DOCUMENT RESUME

ED 337 140 IR 015 127

AUTHOR	Sng, Dennis Cheng-Hong
TITLE	Network Monitoring and Fault Detection on the University of Illinois at Urbana-Champaign Campus Computer Network.
INSTITUTION	Illinois Univ., Urbana. Dept. of Computer Science.
REPORT NO	UILU-ENG-90-1734; UIUCDCS-R-90-1595
PUB DATE	Apr 90
NOTE	120p.; Master's Thesis, University of Illinois at Urbana-Champaign.
PUB TYPE	Dissertations/Theses - Masters Theses (042) -- Computer Programs (101)
EDRS PRICE	MF01/PC05 Plus Postage.
DESCRIPTORS	*Computer Networks; Computer Software; *Computer System Design; *Database Management Systems; Data Collection; *Debugging (Computers); Higher Education; Information Systems; Integrated Library Systems; Systems Analysis; Use Studies
IDENTIFIERS	University of Illinois Urbana Champaign

ABSTRACT

The University of Illinois at Urbana-Champaign (UIUC) has a large campus computer network serving a community of about 20,000 users. With such a large network, it is inevitable that there are a wide variety of technologies co-existing in a multi-vendor environment. Effective network monitoring tools can help monitor traffic and link usage, as well as detect faults and correct them before serious degradation or loss of service to the user community occurs. This report describes a network monitoring and fault detection tool that has been developed and is currently in operation at UIUC. This tool makes use of the Simple Network Management Protocol (SNMP) to collect data from the various network elements monitored and process the data into reports for the network administrator. The report contains discussions of internet subnetting, UIUCnet architecture, motivation and design considerations, data collection and processing, fault detection and isolation, and evaluation, together with comments. The appendices contain a description of the SNMP variables monitored, an SNMPLog, and examples of fault isolation. A glossary of terms and abbreviations is also provided. (52 references) (Author/DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

IR

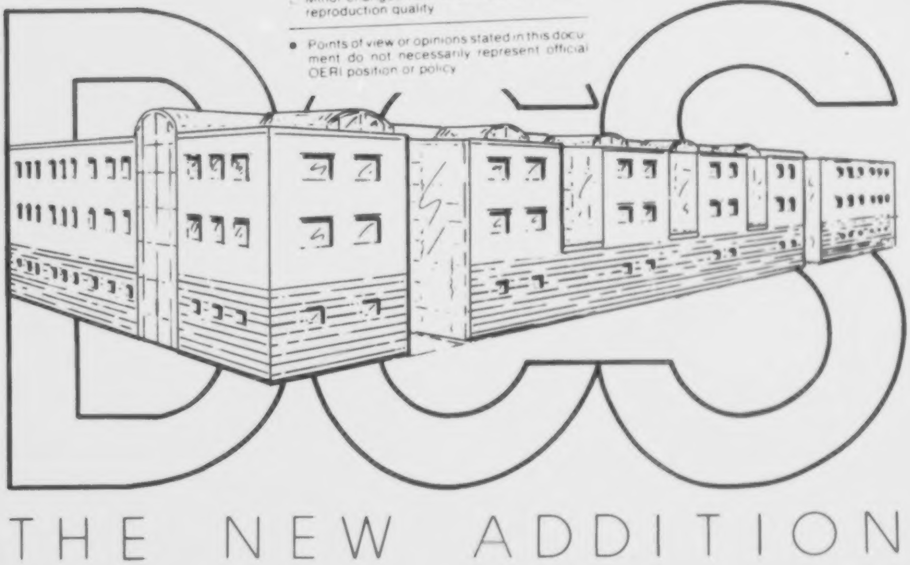
ED337140

IR015127

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it
☐ Minor changes have been made to improve reproduction quality
● Points of view or opinions stated in this document do not necessarily represent official OERI position or policy



REPORT NO. UIUCDCS-R-90-1595

UILU-ENG-90-1734

NETWORK MONITORING AND FAULT DETECTION ON THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN CAMPUS COMPUTER NETWORK

by

Dennis Cheng-Hong Sng

April, 1990

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

William J. Kubitz

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

REPORT NO. UIUCDCS-R-90-1595

NETWORK MONITORING AND FAULT DETECTION ON THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN CAMPUS COMPUTER NETWORK

by

Dennis Cheng-Hong Sng

April 1990

DEPARTMENT OF COMPUTER SCIENCE
1304 W. SPRINGFIELD AVENUE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, IL 61801

NETWORK MONITORING AND FAULT DETECTION ON THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN CAMPUS COMPUTER NETWORK

BY

DENNIS CHENG-HONG SNG

B.Eng. National University of Singapore, 1985

THESIS

Submitted in partial fulfillment of the requirements for the
degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1990

Urbana, Illinois

ABSTRACT

The University of Illinois at Urbana-Champaign (UIUC) has a large campus computer network serving a community of about 20,000 users. With such a large network, it is inevitable that there is a wide variety of technologies co-existing in a multi-vendor environment. Effective network monitoring tools can help in monitoring traffic and link usage, as well as for detecting faults and correcting them before serious degradation or loss of service to the user community occurs.

This paper describes a network monitoring and fault detection tool that we have developed and which is currently in operation at the UIUC. This tool makes use of the Simple Network Management Protocol (SNMP) to collect data from the various network elements monitored and processes the data into reports for the network administrator. We will also give a few examples of faults that have been detected with this tool.

To my father ...

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Ross Veach, the Campus Network Routing Coordinator of the Computer Services Office, UIUC. This thesis would not have been possible without his help, motivation, direction, and support. I would also like to thank my advisor, Tony P Ng, of the Department of Computer Science, UIUC, for his input and guidance.

I would also like to express my thanks to my mother and sister for their encouragement throughout my education. And, finally, I would like to thank Sylvia for her support and companionship as I worked towards this degree.

TABLE OF CONTENTS

1. INTRODUCTION 1

2. INTERNET SUBNETTING 3
    2.1. EVOLUTION 3
    2.2. MOTIVATION 3
    2.3. APPROACHES TO INTERNET ADDRESS ASSIGNMENT 4
        2.3.1. Distinct Networks 4
        2.3.2. Transparent Subnets 4
        2.3.3. Explicit Subnets 5
    2.4. INTERPRETATION OF SUBNETTED ADDRESSES 6
    2.5. SUBNET IMPLEMENTATION WITH ADDRESS MASKS 8
        2.5.1. Hardwired Address Masks 8
        2.5.2. Broadcast-based Protocols 8
    2.6. ROUTING WITH SUBNETS 10

3. UIUCnet ARCHITECTURE 12
    3.1. OVERVIEW OF COMPUTING RESOURCES AND SERVICES 12
    3.2. INTERNAL NETWORK ARCHITECTURE 14
        3.2.1. Design Principles 14
        3.2.2. Campus Backbone 15
        3.2.3. Backbone Routers and Network Interfaces 17
    3.3. EXTERNAL CONNECTIONS 18
        3.3.1. Transit Network uir-uiuc-net 18
        3.3.2. National and Regional Network Connections 18
        3.3.3. Routers on the Transit Network 23
    3.4. PROTOCOLS AND STANDARDS 26

4. MOTIVATION AND DESIGN CONSIDERATIONS 28
    4.1. MOTIVATION 28
    4.2. CENTRALIZED DATA COLLECTION AND MONITORING 29
    4.3. POLLING vs ALERTS 30
        4.3.1. Polling 31
            4.3.1.1. Advantages of Polling 31
            4.3.1.2. Disadvantages of Polling 32
        4.3.2. Alerts 33
            4.3.2.1. Advantages of Alerts 33
            4.3.2.2. Disadvantages of Alerts 33
        4.3.3. Reasons for Selecting Polling 34

5. DATA COLLECTION AND PROCESSING 36
    5.1. SIMPLE NETWORK MANAGEMENT PROTOCOL 36
        5.1.1. SNMP Architecture 36
        5.1.2. Goals of the SNMP Architecture 37
        5.1.3. Representation of Management Information 37
        5.1.4. Operations on Management Information 37
        5.1.5. Protocol Message Exchanges 38
        5.1.6. Management Information Base (MIB) 38

5.2. DATA COLLECTION TOOLS....39
- 5.2.1. Querying Application: snmplog....39
- 5.2.2. Types of Variables Monitored....40
- 5.2.3. Agents Monitored....40

5.3. DATA PROCESSING....42
- 5.3.2. Data Files....42
- 5.3.2. Summaries and Reports....44

6. FAULT DETECTION AND ISOLATION....46

6.1. OVERVIEW....46

6.2. FAULT DETECTION....47
- 6.2.1. Knowledge Base....47
- 6.2.2. Factors Related to Sampling Window....48
  - 6.2.2.1. Window Size....49
  - 6.2.2.2. Static vs Dynamic....49
  - 6.2.2.3. Selection of a Fixed, Sliding Window Scheme....50
- 6.2.3. Fault Reporting Thresholds....50

6.3. FAULT ISOLATION....52
- 6.3.1. Information Required....52
- 6.3.2. Examples of Successful Fault Isolation....53
  - 6.3.2.1. Continuous Restarting of enghall.gw Router....53
  - 6.3.2.2. Badly Formatted Packets on P4100 Routers....55
- 6.3.3. Examples of Less Successful Fault Isolation....56
  - 6.3.3.1. Recurring Restarts of pintail.gw Router....57
  - 6.3.3.2. Input Discards and Frames Lost on eeb.gw Router....58
  - 6.3.3.3. Domain Loop from Indiana....59

7. EVALUATION AND COMMENTS....62

7.1. MONITORING WINDOW....62

7.2. DATA COLLECTION AND PROCESSING....62

7.3. DATA ANALYSIS....63

7.4. ADMINISTRATIVE STRUCTURE....65

7.5. FAULT DISTRIBUTION....66

7.6. EXTENSIONS REQUIRED FOR HOST MONITORING....66

7.7. RELATED WORK....67
- 7.7.1. Network Monitoring Tools....67
- 7.7.2. Artificially Intelligent Tools....69

8. CONCLUSION....71

A. SNMP VARIABLES MONITORED....72

A.1. GENERIC SNMP VARIABLES....72
- A.1.1. System Variables....72
- A.1.2. Interface Table Variables....72

A.2. PROTEON SPECIFIC SNMP VARIABLES....73
- A.2.1. Proteon Generic Variables....73
- A.2.1. Proteon Interface Variables....74
  - A.2.1.1. ProNET-10/80 Interface Variables....74
  - A.2.1.2. Interlan Ethernet Variables....75

A.2.1.3. Proteon/Western Digital Ethernet Variables 76
A.2.1.4. COM-2/COM-4 Serial Interface Variables 77
A.2.1.5. ProNET-4 Token Ring Variables 78

B. SNMPLOG 80
B.1. SNMPLOG UNIX MAN PAGE 80
B.2. SAMPLE SNMPLOG CONFIGURATION FILE 83
B.3. SAMPLE DAILY REPORTS 86
B.3.1. Fault and Traffic Report 86
B.3.2. CICnet Link Usage Report 89
B.4. SAMPLE MONTHLY REPORTS 90
B.4.1. Traffic Report by Day 90
B.4.2. Average Traffic Report by Month 91
B.4.3. Link Usage Report 92

C. FAULT ISOLATION EXAMPLES 94
C.1. DECNET PROBLEMS ON ps.gw 94
C.1.1. Chronological Log of Events 94
C.1.2. Wire Center Physical Connections 94
C.1.3. Input/Output Errors for P4100 Routers in Node 2 96
C.2. DOMAIN LOOP AT INDIANA UNIVERSITY 99
C.2.1. Extracts from the Daily Line Usage Report (Feb 22 & Feb 25) 99
C.2.1. Extracts from the Daily Fault and Traffic Report (Feb 22 & Feb 25) 100

GLOSSARY OF TERMS AND ABBREVIATIONS 101

REFERENCES 106

1. INTRODUCTION

The University of Illinois at Urbana-Champaign has a wide variety of computing resources and a large number of users who make use of these resources. The local users may also need to access resources off campus and, conversely, remote users may need to access resources on campus.

To meet these needs, the university has a large campus computer network to connect its many computing resources. This network provides easy access to local as well as remote resources. Because of it's significant resources (eg. it's supercomputing center) as well as the services (like Domain Name services) that it provides to other networks, the university is very well-connected to other networks, and it is an important site on the Internet.

However, with such a large network and such a large user community, the complexity of the network management problem is increased. Effective network monitoring tools, which can help in fault detection and isolation, become essential. The task of fault detection and isolation is compounded by the fact that there are multiple redundant links between physical networks. Although these redundant links lead to a more reliable network for the user, it makes the fault detection and isolation task more complex.

This paper describes a monitoring tool that we have developed and have been using on the UIUC campus computer network since the fall of 1989. Several components in our network monitoring tool will be discussed here:

(i) Daily reporting of traffic statistics, link usage, and fault statistics and occurrences. These are used in monitoring the day-to-day state of the network.

(ii) Monthly summaries for traffic and link usage. This is used to study longer term trends, and for planning for future expansion.

(iii) Graphs of hourly or daily fault variables for a given interface or router. These are called up on demand to look for patterns in a given parameter, or set of parameters for a given interface or type of interface.

(iv) Data collection of traffic and fault parameters from the monitored network elements. This data is used to support the above-mentioned functions. Archived data is available from September 5, 1989.

In Section 2, we will cover some basics about Internet subnetting, while some background information on the campus network architecture will be presented in Section 3. Section 4 describes our motivation for developing this tool, and some of the design considerations. We will then cover data collection and processing in Section 5. In Section 6, we will describe how the processed data is analyzed, and used to detect and isolate faults. Some examples of the effectiveness of the tool will also be given. An evaluation of our tool, and related work, will be covered in Section 7.

It is assumed that the reader has some basic knowledge of computer networks, the communication protocols involved, and of the Unix operating system. The reader is directed to the following references if some form of review is required: [MeZ90, All89, Kro89, Com88, Tan88]. The world of networks is also a plethora of terms, abbreviations and acronyms. To avoid lengthy explanations of these in the text or in footnotes, a Glossary of Terms and Abbreviations is provided.

2. INTERNET SUBNETTING

2.1. EVOLUTION

Originally, the Internet addressing was viewed as a two-level hierarchy: the top level being the Internet as a whole, and the individual networks forming the level below it. In this model, each host on a network sees its network as a *single entity*.

However, a number of moderately large organizations (eg. universities or companies with more than one building in a site) found this two-level addressing hierarchy inadequate. For administrative or technical reasons, these organizations found it necessary to add a third level to the address hierarchy by dividing their networks into several subnets. These organizations often have to use more than one LAN cable to cover their site. Each LAN can then be treated as a subnet.

2.2. MOTIVATION

An organization may choose to use multiple LANs to cover its campus or site for several reasons [MoP85, Mog84a]:

(i) **Different Technologies**: There may be more than one kind of LAN in use in the organization, eg. a university may have some equipment that support ethernets while other equipment support token rings.

(ii) **Technology Limitations**: Most LAN technologies have some limitations as a result of constraints on some electrical parameters. These, for example, may limit the cable length and the number of hosts that can be connected to a cable.

(iii) **Network Congestion**: Communication among hosts on a network is seldom uniform. Typically, there are one or more groups of hosts that have high mutual communication. These groups of hosts may use up a significant portion of the network bandwidth. Moving these tightly coupled hosts to their own separate networks would ease the congestion.

(iv) **Point-to-Point Links**: Sometimes an organization may have elements located at more than one site. These sites may be too far apart to be connected with LAN technology. Instead, high-speed point-to-point links (eg. T1 lines or microwave relays) connect the LANs at each site.

(v) **Reliability**: Separating a network into subnets enhances the reliability of the overall network by preventing a single malfunctioning hosts or network element from bringing the whole network down. Instead only the relevant subnet is affected while the rest of the network functions normally.

(vi) **Security**: Security can be enhanced by placing nodes to be protected on a separate network and having the communications between them isolated to that subnet rather than broadcasted over the entire network. Thus sensitive traffic can be isolated and protective measures taken, eg. to prevent tapping of the LAN cable.

2.3. APPROACHES TO INTERNET ADDRESS ASSIGNMENT

There are three basic approaches to the assignment of Internet addresses: distinct networks, transparent subnets, and explicit subnets [MoP85, Mog84a].

2.3.1. Distinct Networks

The organization acquires a distinct Internet network number for each cable. The advantage of this approach is that no new or modified protocols are required. The disadvantage is that this approach results in an explosion in the size of the Internet routing tables. This connectivity information is propagated throughout the Internet although it is usually only relevant within the organization. At the time of writing, there are over 1000 networks in the Internet. Some current router implementations do not have enough space to support such large routing tables.

2.3.2. Transparent Subnets

A single network number is acquired for the entire organization, and host numbers are assigned without regard to which LAN a host is on. This requires some protocol that will make the collection of subnets appear to be a single Internet network.

For example, the Address Resolution Protocol (ARP) can translate each Internet address to a hardware address [Com88]. Bridges between the LANs can intercept ARP requests for non-local addresses [Pos84]. However, it may not be possible to do this for all LAN technologies, especially those where ARP protocols are not currently used, or if the LAN does not support broadcasts.

A more fundamental problem is that bridges must discover which LAN a host is on. For example, it could do this using a broadcast algorithm. However, as the number of LANs grows, the overheads associated with broadcasting grows as well. Also, the size of the translation caches maintained in the bridges grows with the total number of hosts on the network.

2.3.3. Explicit Subnets

A single network number is acquired for the entire organization, and the host address space is partitioned by assigning subnet numbers to the LANs. The original approach required a modification of the Internet Protocol, and so only sites using the modified IP implementation can have subnetted networks.

The modifications were relatively minor and maintained compatibility with non-subnetted implementations. Also, it was possible for hosts which believe they were on non-subnetted networks to be used on a subnetted one [Mog84a]. This was useful when it was not possible to modify some of the hosts to support explicit subnetting, or when a gradual transition was preferred.

However, not all networks can support explicit subnets. Some hosts may not have system software to support subnetting. For example, they may be running Unix 4.2 BSD or some other operating system which does not support subnetting, rather than Unix 4.3 BSD which does. Another problem is that IP addresses may have been assigned to hosts in a random manner, without any prior consideration to the possible need for subnetting in the future.

A more efficient solution is to use Proxy ARP [CaQ87] where hosts on a subnetted network can communicate without being aware of the existence of subnets. All subnet handling is done by subnet routers. Proxy ARP will be discussed in Section 2.6.

2.4. INTERPRETATION OF SUBNETTED ADDRESSES

Explicit Subnetting is the most widely used approach because it is the most general and because it has been standardized. Typically, the Internet address for hosts on a subnetted network will take the form:

<network-number> <subnet-number> <host-number>

However, none of the fields above can have the special patterns of all bits set to zero or all bits set to one; these patterns denote "all", as in "all hosts". Thus, with subnetting, we go from the original two kinds of broadcasting (all hosts on a specific network and all hosts on this network) to three [Mog84b]:

(i) **This Physical Network**: A destination address of all zeros or all ones, eg. 0.0.0.0. or 255.255.255.255, means the packet is broadcasted to all hosts on the local physical network and is not forwarded by any router.

(ii) **A Specific Network**: A destination address with a valid network number and all zeros or all ones in the local fields, subnet number and host number, eg. 128.174.0.0 or 128.174.255.255, means the packet is broadcasted to all hosts on the 128.174 network.

(iii) **A Specific Subnet**: A destination address with a valid network number and a valid subnet number, and all zeros or all ones in the host field, eg. 128.174.5.0 or 128.174.5.255, means the packet is broadcasted to all hosts on subnet 5 of the 128.174 network.

However, off wire broadcasts (ie. broadcasts to destinations beyond the subnet of the source host) are seldom allowed in real network implementations. By default, most routers will not forward a broadcasted packet. Thus, in practice, this confines broadcasts to the local network.

As an example of subnetting, consider the Class B network shown in Fig 2.1. This network has a single Class B IP network address, 128.10.0.0, assigned to it. However, it has two physical networks which are treated as subnets. The third octet is used to denote the subnet number and the fourth octet is used for the host number. So hosts on subnet 1 have IP addresses in the form 128.10.1.X, and hosts on subnet 2 have IP addresses in the form 128.10.2.X.

All Internet routers, except the 128.10 network's exterior router, R, treat destinations on that network equally. However, when a packet reaches R, it must be sent to the correct physical network to reach its destination host. R then routes the packet to the correct physical network by examining the third octet of the address [Com88].

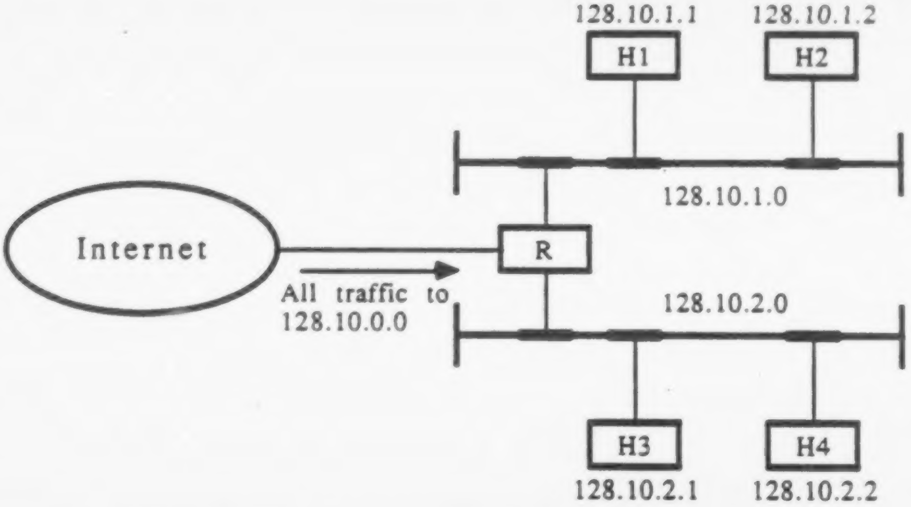


Fig 2.1: Subnet addressing based on third octet of IP address.

The advantage of using this form of hierarchical addressing is that it accommodates large growth because a given router does not need to know as much detail about distant destinations as it does about local ones. One disadvantage is that choosing a hierarchical structure is difficult, and changing an established hierarchy is even more difficult.

The use of subnets is an optional decision made by the site administration. The fact that a network has subnets is transparent to the world outside that network. The subnet implementation is local to the site and can be set up at a site without any global Internet perturbations. The set of subnets is assigned a single IP network number, and appears to the outside world as a single network with that network number. Only within that network does the local structure appear.

The Internet standard for subnet addressing allows each site the flexibility of choosing how to assign its hierarchical addresses. Sites can assign the size of the subnet number field and host number field based on how it expects to grow. So if a site expects to have many networks, it assigns a larger subnet number field, and if a site expects to have many hosts on a network, it assigns a larger host number field. Once a subnet partition has been selected, all hosts on that network must honor it.

2.5. SUBNET IMPLEMENTATION WITH ADDRESS MASKS

A site using subnet addressing must choose a 32-bit *subnet address mask* for each network. Bits in the address mask are set to 1 if the corresponding bit in the Internet address are part of the network and subnet address, and set to 0 if the corresponding bit is part of the host number. For example, the subnet mask

11111111 11111111 11111111 00000000 = 255.255.255.0

specifies that the first three octets of the Internet address identifies the subnet and the last octet specifies the host on that subnet. However, if the mask contains ones in the network number field, and zeros for the rest of the 32-bits, this means that subnets are not in use.

But how does a host determine the address mask of the subnet to which it is connected? There are two basic methods: hardwired address masks and broadcast-based protocols [MoP85, Mog84a].

2.5.1. Hardwired Address Masks

The address mask is hardwired in the host, eg. it may be compiled-in, or (preferably) stored in a disk file. However, this solution is inadequate when there are diskless workstations on a subnet that are bootloaded over the LAN.

2.5.2. Broadcast-based Protocols

Broadcast-based protocols are preferred since most LAN technologies support broadcasting. Ideally, a host could obtain its subnet address mask together with other configuration that it acquires when it is booted. Alternatively, it can make a special broadcast request for its subnet address mask.

A newly-booted host usually needs to collect information like its IP address, the hardware address of its router, and the IP address of the domain name server, as well as the subnet address mask. It would be more economical to acquire all this information in a single request, rather than doing numerous broadcasts and adding to the overhead on the network. The mechanisms used to boot diskless workstations can also be used to load host specific configuration files that contain all the required information [CrG85].

However, if it is necessary to make a separate request for the subnet address mask, the ICMP protocol [Com88] provides the Address Mask Request and Address Mask Reply message types. A host broadcasts an Address Mask Request message on the subnet. A router connected to the subnet that receives this message will respond with an Address Mask Reply to the requesting host, or if the address of the requesting host is unknown (eg. the request message had a null source address), then the reply is broadcasted as well.

But there are problems with badly configured hosts announcing the wrong information. The requesting host will take the mask given by the first response as the subnet's address mask. So, if a badly configured host replies with the wrong address mask, then the requesting host will use the wrong address mask in all its communications.

Since there is only one possible address mask for each LAN, there is no problem if more than one router responds. Also, it is assumed that hosts reboot infrequently, and so requests for address masks are infrequent. Thus the broadcast overhead from the use of such a protocol is small.

What happens if a host is unable to get a response to its Address Mask Requests, even after several attempts? This could happen in the following situations:.

(i) The physical net exists in (permanent) isolation from all other nets. In this case, the subnet field width is zero.

(ii) Subnets are not in use on this network, and no host supports the ICMP Address Mask Request packet type. The subnet field width is zero too.

(iii) All routers on the local net are (temporarily) down. There is no way to determine the subnet address mask, so assume that the subnet field width is zero. When one of the routers on the physical network comes up, it should broadcast an Address Mask Reply with the appropriate mask. The hosts on the physical network that have incorrect subnet address masks can then update their masks.

2.6. ROUTING WITH SUBNETS

In subnetted networks, there is another question in the routing problem: Is the destination host on the same physical network, rather than the same Internet network, as the source host? The source host sends the packet directly to the destination host in the first case, and to the subnet's router in the second case. Proxy ARP, defined in RFC-1027 [CaQ87], allows hosts on a subnetted network to communicate without being aware of the existence of subnets.

Suppose a packet is to be addressed to a destination host on the same physical network as the source host. The ARP server module in the destination host, on receipt of the ARP request, checks the destination IP address and discovers that it corresponds to its own IP address. The destination host then sends the requesting host an ARP response giving it's hardware address. The source host then caches this IP address to hardware address mapping for use in future communications with the destination host.

Suppose a packet is to be addressed to a destination host on the same network, but on a different subnet from the source host. The ARP server module in the subnet's router, on receipt of the ARP request, checks the destination IP address to see if it (the router) knows of a route to the destination host. If it does, the router sends the requesting host an ARP response giving its own hardware address. The requesting host then caches a mapping of the destination host's IP address to subnet router's hardware address, and sends packets to the router, which then forwards the packet in the usual manner.

Suppose the host sends out an ARP request for an IP broadcast address. In this case, the subnet router should not reply. If it were to respond with an ARP reply, it would result in the requesting host transmitting traffic with a broadcast address. This could result in every host on the subnet replying to the source host, causing a broadcast storm.

Furthermore, a subnet router must not issue an IP broadcast using the source subnet broadcast address (eg. 128.174.5.255 for the subnet number 5 on the 128.174 network). Hosts on the physical network that receive the broadcast will not understand that it is a broadcast address since they either do not have a subnet implementation or do not have subnets enabled. Instead, the router should use the network broadcast address (ie. 128.174.255.255 for the example above).

Using Proxy ARP, all subnet handling is done in the subnet routers. No changes need to be made to the normal ARP protocols or routing schemes in the hosts. The hosts are not aware of subnets and see their physical networks as one big IP network. If the host does have an implementation that is aware of subnets, then it can have its subnet implementation enabled. Otherwise, if the host does not understand subnets, then its address masks should be set to cover only the IP network number, excluding the subnet bits, and let the router handle the subnets for it using Proxy ARP.

Network and routing administrators must ensure that the address masks are the same on all the subnet routers. A common error is to set the address mask on a host without a subnet implementation to include subnet bits. This causes the host to fail in its attempts to send packets to hosts not on its local subnet.

Finally, there may be situations where there are multiple routers between two physical networks, perhaps for better reliability or because there is a large amount of traffic between the two networks. A requesting host will use the first ARP response it receives. In this way, load balancing among the parallel routers is achieved since the least loaded router is most likely to respond first.

3. UIUCnet ARCHITECTURE

3.1. OVERVIEW OF COMPUTING RESOURCES AND SERVICES

The UIUC has several large mainframes, and about 3000 minicomputers, workstations, and personal computers, spread out over a 700 acre campus. These form a mixed environment of hardware from a variety of vendors, using a variety of protocols. The campus computer network, UIUCnet, connects these machines. The university's Computing Services Office (CSO) serves as a developer and manager of the campus network facilities.

The CSO manages the campus's main computing resources which include a Convex C220 mini-supercomputer, four IBM mainframes, and two Sequents. It also operates other resources like a pair of public dial-up terminal servers, a tape server, a print server, a pair of Domain Name servers, a pair network time servers, and electronic mail gateways.

In addition to the central computing resources under the CSO, the various departments also have a wide range of resources of their own. For example, the Computer Science Department operates three Encore Multimaxes, Sun workstations, AT&T 3B2s and 3B20s, IBM RTs, and other machines spread out over several sites. The National Center for Supercomputing Applications (NCSA) operates two supercomputers, a Cray X-MP and a Cray-2, as well as numerous Sun workstations and Apple Macintoshes. The Center for Supercomputing Research and Development (CSRD) operates several Alliant FX/8 mini-supercomputers, numerous Sun workstations and SPARCstations, and several Apollo workstations.

A good example of a widespread service provided to both the local and remote user community is the Domain Name service [Com88] provided by the pair of Domain Name servers. These two machines provide primary Domain Name service for the campus. If both these machines are down, secondary Domain Name service is provided by a backup nameserver at Indiana University. All three nameservers are authoritative for the campus.

In addition to providing primary Domain Name service to the campus, the twin nameservers also provide primary Domain Name service to several external networks that do not have their own nameservers. These include the networks of Bradley University, Millikin University, and the US Army Construction Engineering Research Laboratory. They also provide backup Domain Name service in the manner that Indiana University provides such a service to UIUC. Backup service is provided to Indiana University, Northern Illinois University, Ohio University, and several other external networks.

The UIUCnet provides the campus's 20,000 users with access to all these resources and services, as well as to computers located across the nation and overseas. Through UIUCnet, campus users have direct access to several international, national and regional wide-area networks (WANs) like the NSFnet, and the CICnet, and MRnet regional networks. These external networks perceive the entire campus as a single Class B network.

Within the campus, there are four Proteon token rings, two HYPERchannels, about 150 Ethernets, about a dozen IBM Token Rings, and many more Appletalk local area networks (LANs), as well as low-speed phone lines and Sytek LocalNet networks. The four Proteon rings form the major part of the campus backbone network. The communication media ranges from twisted-pair lines to coaxial cable for LANs, and about 20 miles of multi-fiber optical cables for the backbone. Over 75 buildings are linked by more than 80 routers to the backbone.

UIUCnet is a Class B network with network name *uiuc-campus-b* and address 128.174.0.0. This network covers the UIUC academic and administrative departments, as well as the NCSA, the CSRD and Beckman Institute for Advanced Science and Technology. Because of the large number of local networks on the campus, explicit subnetting is used. The third octet of the IP address is used as a subnet number. The subnet address mask is thus 255.255.255.0.

3.2. INTERNAL NETWORK ARCHITECTURE

3.2.1. Design Principles

The guiding principle is to provide the network users with easy access to a basic service at a uniformly high level of performance. This is achieved, physically, by having a high-speed backbone network with subnets branching off at each campus building.

The backbone consists mainly of four fiber-optic star-shaped token rings, with their wiring centers located near the four corners of the campus. Two of the rings are 80 Mbps ProNET-80 rings, which mainly serve the engineering and sciences campus, while the other two are 10 Mbps ProNET-10 rings. These rings partially overlap, but the two ProNET-80 rings and one of the ProNET-10 rings come together on an ethernet in the Digital Computer Laboratory (DCL), the home of the Department of Computer Science and the CSO.

The purpose of the backbone is to provide access to as many of the computing resources as possible with emphasis on host to host communication. High throughput is necessary to handle fast data transfer between hosts. The token ring access method is contention free, which means the response time is maintained even at high traffic loads. The benefits of using optical fiber are: immunity to electromagnetic interference (EMI), which results in high reliability, and high bandwidth, which gives good throughput.

Another design principle is to use routers, rather than bridges, to link subnets to the backbone. The reasons for this choice are mainly:

(i) The size of the network in terms of the number of subnets and hosts as well as the geographical distances involved.

(ii) The greater functionality of routers compared to bridges.

From these routers, smaller building-wide or department-wide backbone networks, or simply individual LAN segments, branch off the main campus backbone. For example, both the NCSA and the Beckman Institute each have their own HYPERchannel for a backbone, while the Dept of Computer Science uses an ethernet. Most of the routers connected directly to the backbone come under the administrative control of the CSO. The routers that connect the LANs within the departmental networks fall under the administrative control of the departments.

3.2.2. Campus Backbone

The campus backbone consists of four interconnected star-shaped token rings and an ethernet which links the rings to the outside world. There are also about half a dozen 56 kbps serial lines to some local sites that do not have fiber pulled up to them. The network names and addresses of the backbone rings and ethernet are given below:

Network Name	Address	Type of LAN
uiuc-narya-net	128.174.100.0	ProNET-80
uiuc-nenya-net	128.174.55.0	ProNET-80
uiuc-vilya-net	128.174.7.0	ProNET-10
uiuc-campusnet	128.174.3.0	ProNET-10
uiuc-bone-net	128.174.89.0	Ethernet

The backbone rings are redundantly connected to each other and to *uiuc-bone-net* as shown in Fig 3.1. Each ProNET-10 ring is connected to three of the backbone redundancy routers, while each ProNET-80 ring is connected to four of the backbone redundancy routers. Also, *uiuc-bone-net* is connected to the backbone rings via two of the backbone redundancy routers.

These multiple redundant links result in increased reliability, and ensure that the backbone is able to provide a consistently high level of connectivity and service with minimal down time. Each link is capable of handling all the inter-subnet traffic. Thus, failure of any link will not affect the connectivity or service significantly as there will be at least one other link to the other subnets.

An additional factor of reliability is that one of the rings (the .100 subnet) actually consists of a dual ring connection between adjacent wire centers. The secondary ring is not used to carry network traffic during normal operation. However, if the primary ring is severed or malfunctions somewhere between adjacent wire centers, the secondary cable can be used to re-route traffic. This pair of rings are counter-rotating, because the messages on the secondary ring moves in the opposite direction to the primary ring [Ged89].

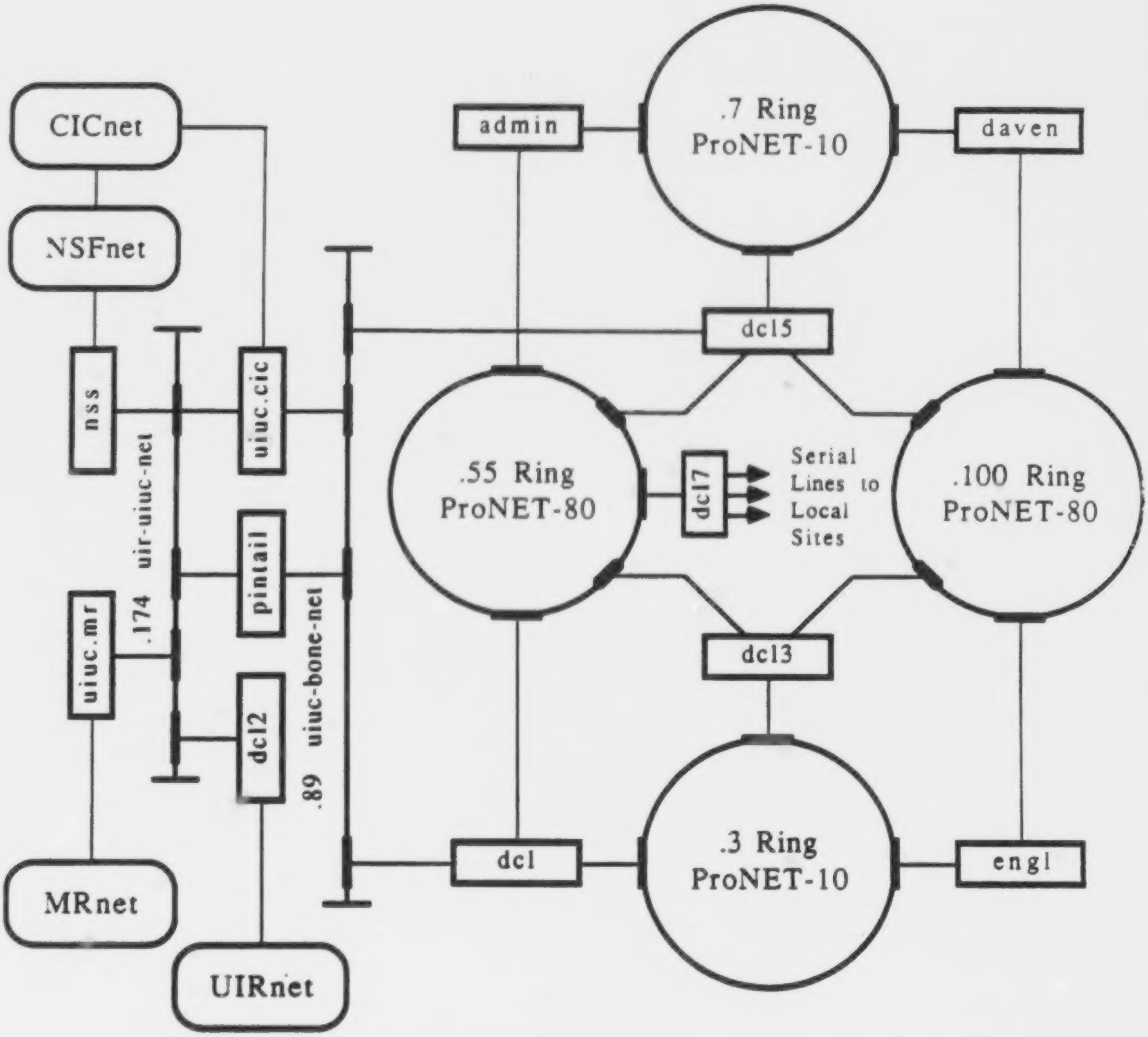

CICnet
NSFnet
MRnet
UIRnet
nss
uiuc.cic
uir-uiuc-net
.174
pintail
uiuc.mr
dcl2
uiuc-bone-net
.89
admin
.7 Ring
ProNET-10
daven
dcl5
.55 Ring
ProNET-80
dcl7
Serial
Lines to
Local
Sites
.100 Ring
ProNET-80
dcl3
dcl
.3 Ring
ProNET-10
engl

Fig 3.1: **The Campus Backbone with Internal Redundancy and External Links**

Many of the CSO mainframes (the Convex, uxh; the Sequent, ux1; and the IBM 3081, vmd) are attached to *uiuc-cso-dclnet* (128.174.5.0). Other important machines are the CSO nameserver (garcon), the campus mail server and backup nameserver (uxc), and the dialup terminal servers (mossberg and term1). Both garcon and uxc are DEC VAXstation 3500s, while the terminal servers are CISCO ASMs. The terminal server mossberg has 32 dial-up lines while term1 has 16 dial-up lines (which will later be upgraded to 32 lines).

3.2.3. Backbone Routers and Network Interfaces

Most of the routers connected to the backbone are Proteon Inc. P4100 and P4200 IP routers. The P4100 has an Intel 80286 CPU, and is basically an IBM PC/AT clone running router software. These are mainly used to connect to the ProNET-10 rings.

The higher performance P4200 is based on the Motorola 68010 and 68020 microprocessors, with 0.5 to 2.0 MB of memory (usually depending on the number of protocols they have to run). These are mainly used to connect to the ProNET-80 rings. In particular, the redundancy routers (admin, daven, dcl, dcl3, dcl5, and engl) have 68020 CPUs and will be upgraded to 2 MB of memory since, in their role as routers interconnecting the backbone rings, they have to switch a higher number of packets and support a complete set of protocols. The dcl7 router, which connect the serial lines to remote sites to the backbone rings, also has a 68020 CPU.

The Network Systems Corp (NSC) EN641 routers are high performance routers which can support up to a dozen ethernet interfaces together with a HYPERchannel interface, in a single box. There are seven such routers on the campus network, three on each of the HYPERchannels which form the departmental backbones for NCSA and Beckman Institute, and one, pintail, the campus's exterior router onto *uir-uiuc-net*.

The CICnet and MRnet routers at UIUC are Cisco AGS and CGS machines respectively. Both these models have the same CPU, but the AGS model has a larger card cage, and so is capable of having more plug-in memory or interface modules. These routers have T1 lines to their regional backbones.

There are several types of network interface found in these routers. The most numerous are Ethernet interfaces: Interlan, Proteon, and Western Digital, with the Interlan interfaces being the oldest. These connect to the departmental ethernets. The next most common are the ProNET-10 and ProNET-80 optical fiber token ring interfaces, which are used to connect to the backbone rings. There are also HYPERchannel interfaces, ProNET-4 IEEE 802.5 token ring interfaces, and 56 kbps and T1 serial lines.

The Proteon routers can have only one network interface per plug-in card. This means that each packet switched by a Proteon router must pass through the system backplane. The NSC EN641s and Cisco routers support several network interfaces per card. If a packet is routed from an interface on a card to another interface on the same card, then it need not go through the system backplane. Consequently, the EN641s and Cisco routers, if they are properly configured, are usually capable of higher packet switching rates than the Proteon routers.

3.3. EXTERNAL CONNECTIONS

3.3.1. Transit Network *uir-uiuc-net*

The network *uir-uiuc-net* (130.126.174.0) connects UIUCnet to the outside world via UIUCnet's exterior router, pintail. This network plays an important role in the Internet because the NSFnet, and regional networks CICnet, MRnet, and UIRnet, all have one of their backbone routers connected to it.

Almost all of this network's traffic involves one remote location. Since, many networks make use of *uir-uiuc-net* as a transit point to other destinations, a significant portion of this network's traffic involves two remote locations. Because of its role as a transit network, it is the busiest network monitored in this study. For more detailed information on the traffic patterns and an analysis of the traffic on the subnet *uir-uiuc-net*, refer to [Sng89].

3.3.2. National and Regional Network Connections

UIUC is a major site for NSFnet, CICnet and MRnet. These networks all have one of their backbone routers located on the transit network *uir-uiuc-net*.

These routers are remotely managed by their respective network operations centers, with the UIUC providing infrastructural support like space, power, and air-conditioning.

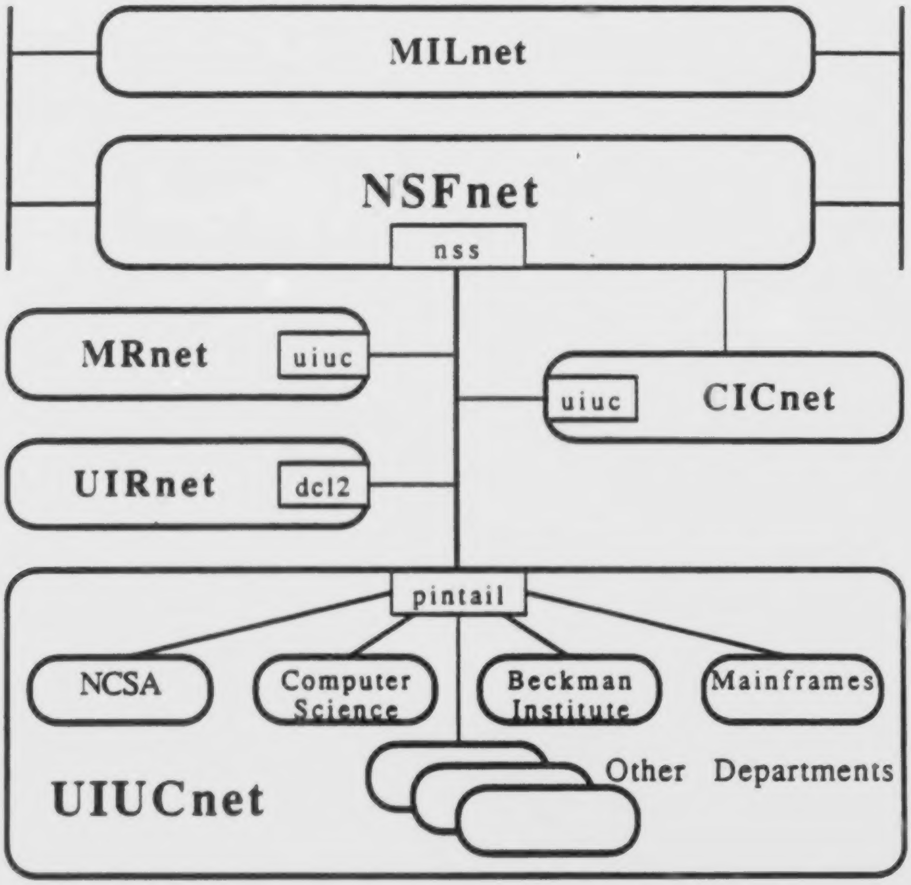
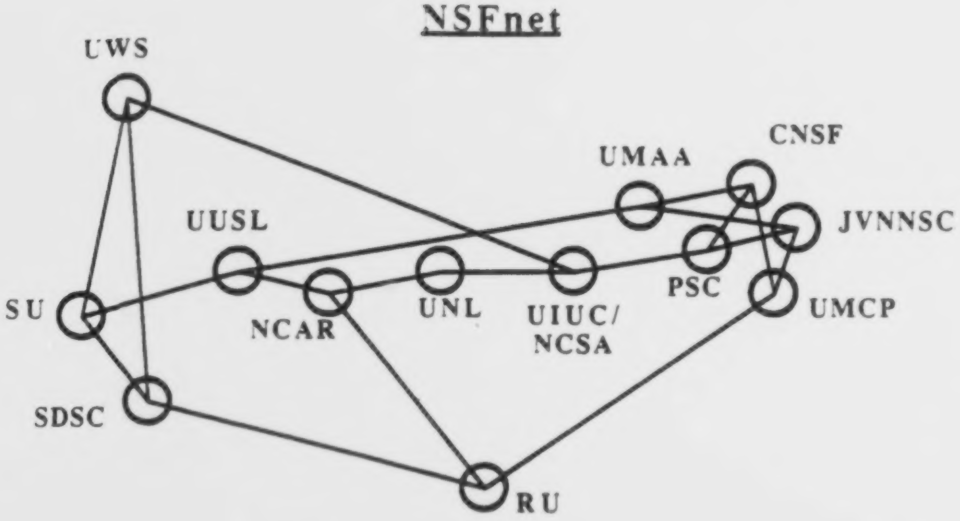

MILnet
NSFnet
nss
MRnet
uiuc
CICnet
uiuc
UIRnet
dcl2
pintail
NCSA
Computer Science
Beckman Institute
Mainframes
Other Departments
UIUCnet

Fig 3.2: External Connections of UIUCnet

The **NSFnet** is sponsored by the National Science Foundation (NSF), and it consists of a cross-country backbone connecting 13 nodes, a set of regional networks, and a set of over 1000 campus networks (at the time of writing) [Com88, Red89]. Six of the backbone nodes are NSF supercomputing centers (the first being UIUC), while the others are major nodes on the regional networks. This three-level hierarchy of networks allows researchers to communicate with each other either at a campus, regional, or national level [Com88]. The NSFnet is a class B network with IP address 129.140.0.0.

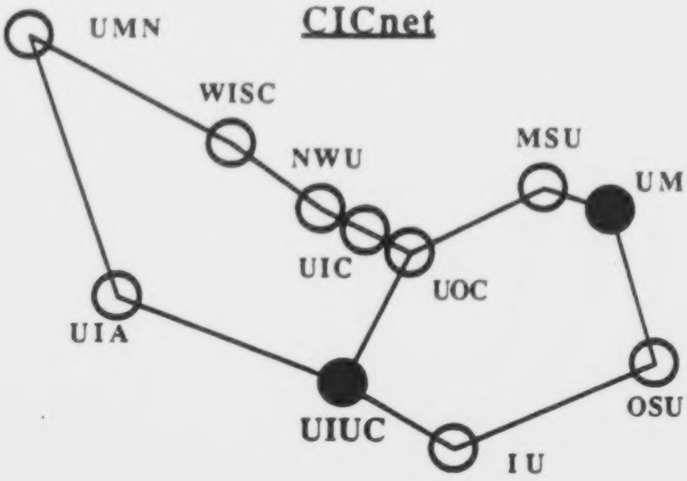


Abbreviation	NSFnet Backbone Sites (Nodal Switching Stations)	Associated Regional
UWS	Univ of Washington, Seattle WA	NORTHWESTNET
SU	Stanford Univ, Palo Alto, CA	BARRNET
SDSC	San Diego Supercomputer Center, San Diego, CA	SDSCNET, CERFNET
UUSL	Univ of Utah, Salt Lake City, UT	WESTNET
NCAR	Nat'l Center for Atmospheric Research, Boulder, CO	WESTNET, USAN
UNL	Univ of Nebraska, Lincoln, NE	MIDNET
RU	Rice Univ, Houston, TX	SESQUINET, THENET
UIUC/NCSA	Univ of Illinois/Nat'l Center for Supercomputing Applications, Urbana-Champaign, IL	CICNET, MRNET
UMAA	Univ of Michigan, Ann Arbor, MI	CICNET, MERIT
PSC	Pittsburgh Supercomputing Center, Pittsburgh, PA	PSCNET
CNSF	Cornell Nat'l Supercomputer Facility, Ithaca, NY	NYSERNET, ONET
JVNNSC	John von Neumann Nat'l Supercomputer Center, Princeton, NJ	JVNCNET, NEARNET
UMCP	Univ of Maryland, College Park, MD	SURANET

Fig 3.3: NSFnet Backbone Network

Since the NSFnet is a major production network, rather than a research network, it is crucial that it does not get partitioned or goes down. Each backbone node is connected to at least three other nodes by means of T1 (1.544 Mbps) lines, although there are proposed upgrades to T3 (44.736 Mbps) lines in the future. This redundancy ensures that faults in any of the links will not isolate a particular node, thus improving reliability. There is also an elaborate network monitoring station at Merit, a consortium of eight state-supported universities in Michigan, based in Ann Arbor [Red89].

There are over a dozen NSF funded regional networks that span almost every state. Each regional consists of about a dozen universities that are within the same geographic area. The NSF provides seed money to cover the initial costs of setting up these regional networks, but later leaves them to operate independently.

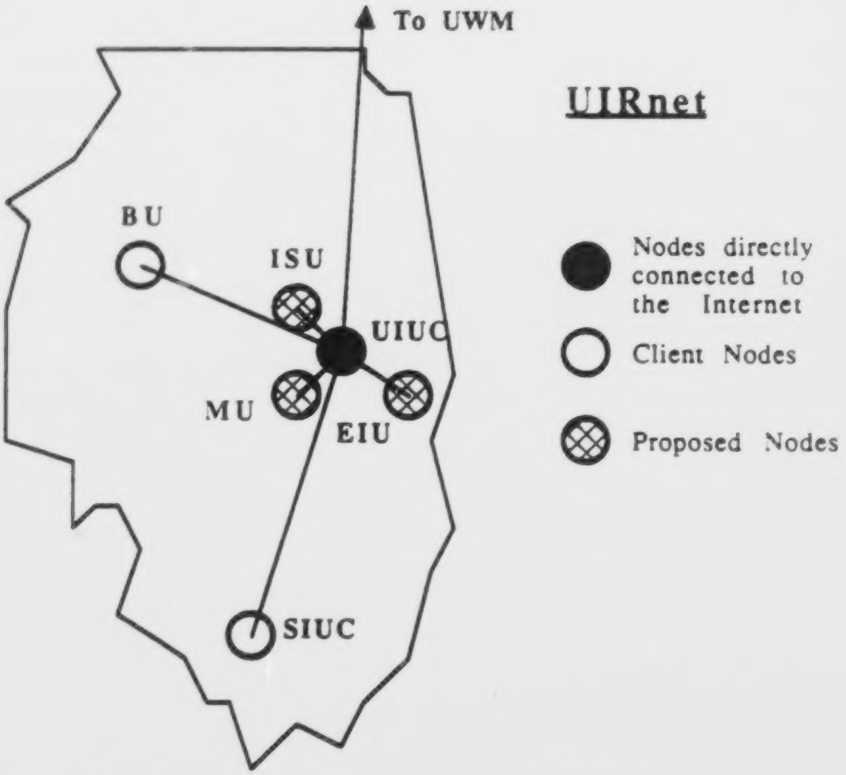


Abbreviation	Institution	Location
UIUC	University of Illinois	Urbana-Champaign, IL
UIA	University of Iowa	Iowa City, IA
UMN	University of Minnesota	Minneapolis-St Paul, MN
WISC	University of Wisconsin	Madison, WI
NWU	North Western University	Evanston, IL
UIC	University of Illinois	Chicago, IL
UOC	University of Chicago	Chicago, IL
MSU	Michigan State University	East Lansing, MI
UM	University of Michigan	Ann Arbor, MI
OSU	Ohio State University	Columbus, OH
IU	Indiana University	Bloomington, IN

Fig 3.4: CICnet Backbone Network

The campus network component of NSFnet were funded as experiments of new technologies or to provide supercomputer access for specific research groups. Each supercomputer center includes a consortium of research groups that are connected to it via leased lines.

CICnet is a regional network administered by the Committee of Institutional Cooperation (CIC) and it connects the Big-Ten universities in the mid-west plus the University of Chicago [Fig 3.4]. It is a class B network with IP address 131.103.0.0.


To UWM
UIRnet
BU
ISU
UIUC
MU
EIU
SIUC
Nodes directly connected to the Internet
Client Nodes
Proposed Nodes

Abbreviation	Institution	Location
BU	Bradley University	Peoria, IL
EIU	Eastern Illinois University	Charleston, IL
ISU	Illinois State University	Normal, IL
MU	Milliken University	Decatur, IL
SIUC	Southern Illinois University	Carbondale, IL
UIUC	University of Illinois	Urbana-Champaign, IL
UWM	University of Wisconsin	Milwaukee, WI

Fig 3.5: UIRnet - Current and Proposed Connections

Each node on the CICnet backbone is connected to at least two other nodes with T1 lines. The CICnet is connected to the NSFnet at two sites, one being at UIUC and the other at the University of Michigan. UIUC is the primary default site handling CICnet-NSFnet traffic from about 75% of the sites while Michigan handles the rest. UIUC is connected to the University of Iowa, University of Chicago and Indiana University.

MRnet, the Minnesota Research network, is the regional network connecting the Minnesota Supercomputing Center and about a dozen northern mid-west universities. Also connected to the MRnet are private corporations like Cray Research Inc. It is a Class B network with address 137.192.0.0. The MRnet is connected to the NSFnet at UIUC. Traffic from MRnet gains access to its backbone router on *uir-uiuc-net* via a T1 line.

The Class B network **UIRnet** (the University of Illinois regional network) provides the university with better administrative control of its external connections. It has the network name *uiuc-ncsa* and network number 130.126.0.0. At present, UIRnet provides external connections to the Internet with direct connections to the NSFnet, and its regional networks (eg. CICnet and MRnet) at UIUC. UIRnet clients include Bradley University, Southern Illinois University at Carbondale, the University of Wisconsin at Milwaukee, and the US Army Corps of Engineers' Construction Engineering Research Laboratory.

3.3.3. Routers on the Transit Network

There are several backbone routers belonging to networks of the Internet that are connected to the transit network *uir-uiuc-net*, and one of them, together with the UIUCnet's exterior router, is connected to *uiuc-bone-net* as well [Fig 3.6]. The campus's exterior router and UIRnet's backbone router are administered by the CSO, while those owned by NSFnet, CICnet, and MRnet are administered by their own relevant bodies. The router protocols used by these routers are either the External Gateway Protocol (EGP) [Com88], or the Routing Information Protocol (RIP) [Com88], or both.

The backbone router of the NSFnet at UIUC is **nss.gw.uiuc.edu**. All traffic to NSFnet from UIUCnet, UIRnet, and MRnet, and most of the traffic from CICnet, will pass through this router. This router is linked to the rest of NSFnet via three T1 lines. This router only participates in the EGP exchange and does not listen to any of the RIP broadcasts.

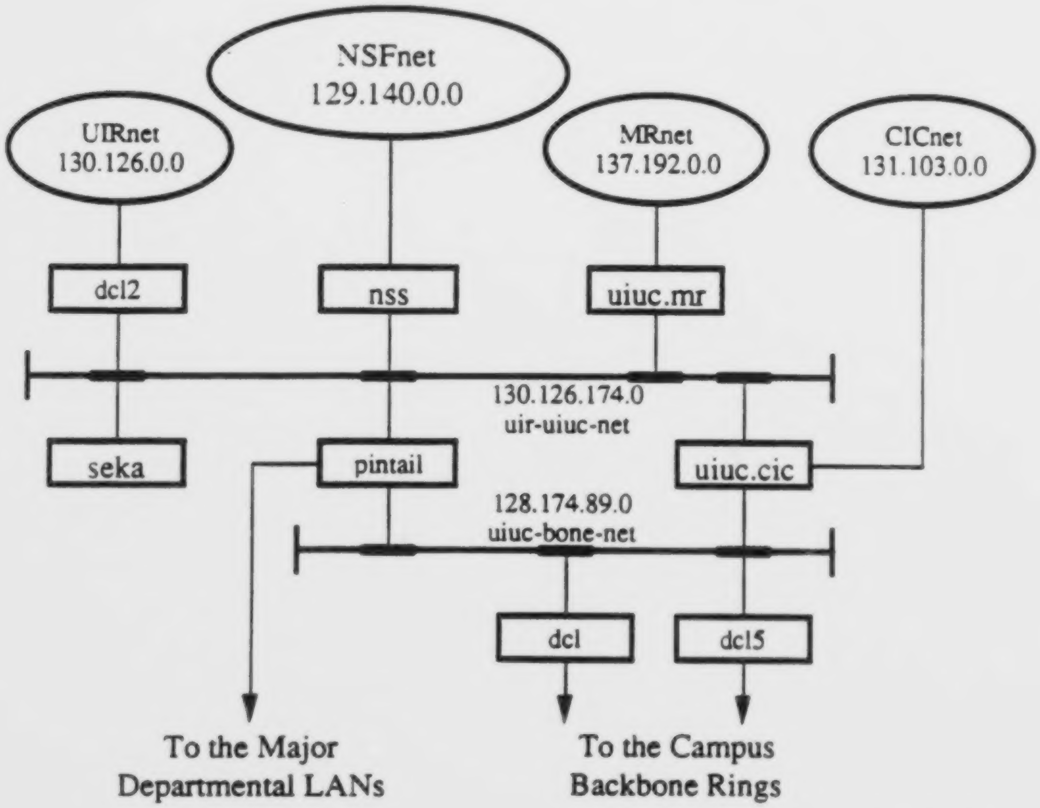

NSFnet
129.140.0.0
UIRnet
130.126.0.0
MRnet
137.192.0.0
CICnet
131.103.0.0
dcl2
nss
uiuc.mr
130.126.174.0
uir-uiuc-net
seka
pintail
uiuc.cic
128.174.89.0
uiuc-bone-net
dcl
dcl5
To the Major
Departmental LANs
To the Campus
Backbone Rings

Fig 3.6: Exterior routers on *uiuc-bone-net* and *uir-uiuc-net*.

The router nss.gw.uiuc.edu actually consists of nine IBM PC/RTs, which make up the Nodal Switching System (NSS) [Red89]. The RTs are connected by dual token rings, one for operational use and the other for maintenance/debugging purposes. The three T1 lines are connected to three of the RTs which are called the Packet Switching Processors (PSP). The RT that interfaces to the regional network, in this case the ethernet *uir-uiuc-net*, is called the Exterior Packet Switching Processor (E-PSP). Finally, one RT is designated the Routing Control Processor (RCP) which is the controller node for the NSS. The other RTs are used for data logging purposes or as backups.

The backbone router of CICnet at UIUC is **uiuc.cic.net**, a CISCO AGS router. It connects UIUCnet to the rest of CICnet (via three T1 lines) and CICnet to NSFnet (via *uir-uiuc-net* and router nss). This router advertises the lowest default so all of the other routers that do not participate in the EGP exchange will send their external traffic to it. It will then forward the packets accordingly.

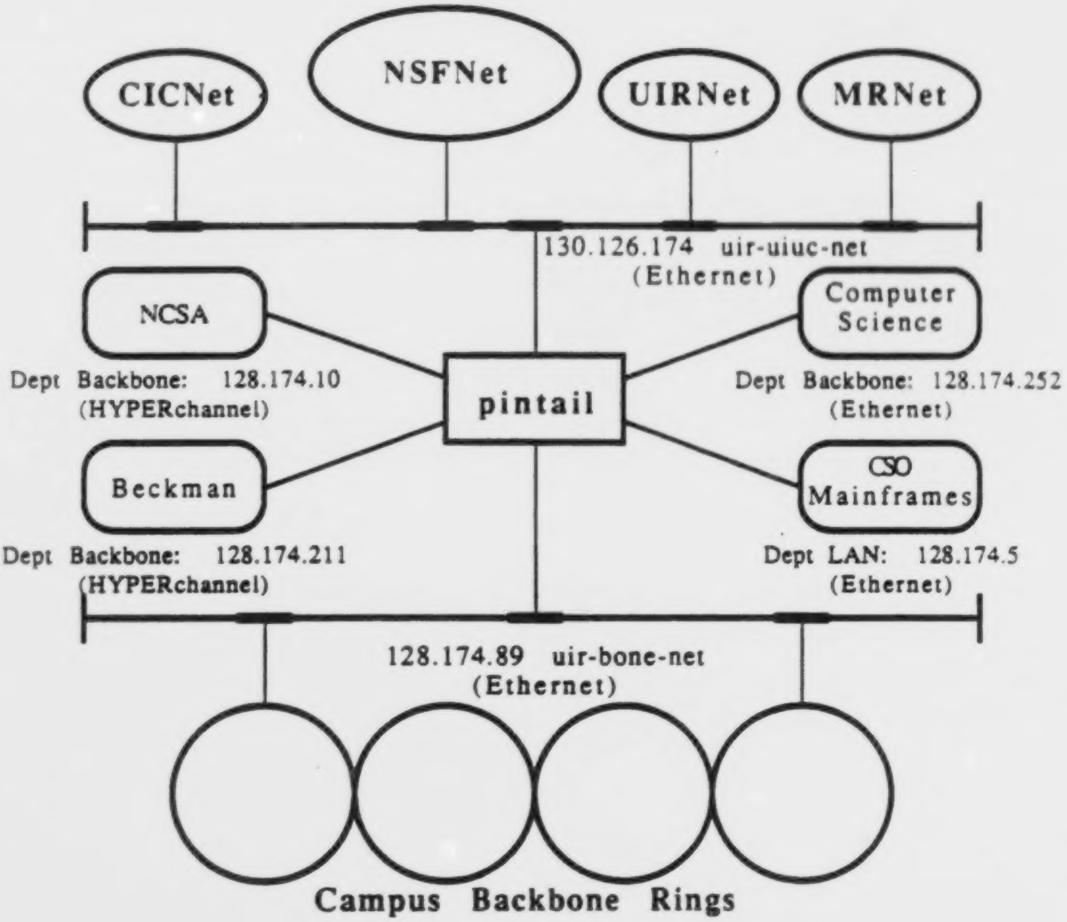

CICNet
NSFNet
UIRNet
MRNet
130.126.174 uir-uiuc-net
(Ethernet)
NCSA
Computer Science
Dept Backbone: 128.174.10
(HYPERchannel)
pintail
Dept Backbone: 128.174.252
(Ethernet)
Beckman
CSO Mainframes
Dept Backbone: 128.174.211
(HYPERchannel)
Dept LAN: 128.174.5
(Ethernet)
128.174.89 uir-bone-net
(Ethernet)
Campus Backbone Rings

Fig 3.7: Interfaces of UIUCnet's Exterior Router pintail.gw.uiuc.edu.

The backbone router of MRnet at UIUC is **uiuc.mr.net**, a CISCO CGS router. It connects to the rest of MRnet (via a T1 line), and to the NSFnet (via *uir-uiuc-net* and router nss).

The backbone router of UIRnet at UIUC is **dcl2.gw.uiuc.edu**, a Proteon P4200. It connects Bradley University, the University of Wisconsin at Milwaukee, and USA-CERL to UIRnet via 56 kbps lines. It also has an interface to a ProNET-10 token ring (which also forms part of UIRnet) over which traffic to and from the Southern Illinois University at Carbondale flows.

The router **pintail.gw.uiuc.edu**, a NSC EN641, is the exterior router of UIUCnet, and by default, handles most of the traffic between *uir-uiuc-net* and UIUCnet. It has interfaces to *uiuc-bone-net* (and, hence, to the backbone rings), as well as to UIUCnet's four busiest departmental networks: the NCSA's backbone, the Beckman Institute's backbone, the Department of Computer Science's backbone, and the CSO's mainframe ethernet [Fig 3.6].

The SUN 3/50 workstation **seka.cso.uiuc.edu** does not switch any packets. It only listens to RIP and helps in the EGP announcements. It also serves as the platform for the network monitoring software used in this study.

Only the routers that connect directly to the UIUCnet, like pintail.gw.uiuc.edu and uiuc.cic.net, know that subnetting is used on the campus network. The routers nss.gw.uiuc.edu and uiuc.mr.net only know that UIUCnet is a single class B network 128.174.0.0.

3.4. PROTOCOLS AND STANDARDS

Here, the major protocols and standards supported on the UIUCnet will be highlighted. The UIUCnet supports a variety of protocols, the most significant ones being:

(i) **Transmission Control Protocol/Internet Protocol (TCP/IP)**: This runs on the largest number of systems and networks. These are typically Unix systems running the Berkeley Software Distribution. This provides remote login, electronic mail, file transfer, shared network resources, user transparent network topology, redundant and dynamic configuration, and network management tools. This is readily inter-connected with products from other vendors.

(ii) **DECnet**: There are a number of DECnet networks on campus with campus-wide and national connections. The DECnet network provides services (remote login, etc) similar to the TCP/IP network. Interfaces to other protocols, like TCP, are available.

(iii) **IBM**: There are a variety of IBM communication protocols for host-to-host and host-to-terminal communication. These are generally not compatible with equipment from other vendors.

At present, all the routers on the campus support TCP/IP, and several (about two dozen) also run DECnet simultaneously.

The following standards have been adopted for UIUCnet:

(i) TCP/IP **File Transfer Protocol** (**FTP**) and **Trivial File Transfer Protocol** (**TFTP**) are the campus file transfer standards.

(ii) **Simple Mail Transfer Protocol** (**SMTP**) is the campus mail standard.

(iii) **Mail gateways** support ARPAnet, BITnet, CSnet, DECnet and UUCP.

(iv) TCP/IP **TELNET** is the minimum remote login standard although **RLOGIN** is made available on many hosts.

(v) The **BIND** nameserver software provides host name services to the campus and remote networks.

(vi) The standard **/etc/hosts** and **/etc/networks** files are maintained on most TCP/IP implementations.

(vii) The **Routing Information Protocol** (**RIP**) is the current internal router protocol.

(viii) The **Exterior Gateway Protocol** (**EGP**), **HELLO** and **RIP** are the routing protocols used to interface with external networks.

(ix) The **Simple Network Management Protocol** (**SNMP**) is the current network management protocol.

The emphasis on using the TCP/IP protocols and the accompanying standards is because they are in wide use on the Internet, and because they are well-defined and well-supported by the industry. Also, TCP/IP provides a good migration path to the International Standardization Organization (ISO) Open System Interconnect (OSI) model standards when they become established in the future.

4. MOTIVATION AND DESIGN CONSIDERATIONS

4.1. MOTIVATION

Because of all the redundancy built into the network to increase its reliability, the failure of a network element may not be noticeable to the user or network administrator. However, when a problem becomes noticeable, there is a good chance that several failures have occurred. For example, any one of the backbone redundancy routers can handle all the traffic between the rings it is connected to. For a backbone ring to be isolated, all the redundancy routers between it and the other rings have to fail. This will result in a disruption of service to a large portion of the user community.

With such a large network as UIUCnet, manual monitoring of the network parameters would be time consuming, tedious, and error prone. Thus, it is important for the network administrator to have network monitoring tools so as to have the necessary data to characterize the network's performance, and to be aware of faults before they lead to serious network downtime. These tools should also be able to provide historical data so that he can perform trend analysis, and plan for future network expansion, or schedule preventive maintenance before failures occur.

Hence, network monitoring tools help the network administrator in two of his management tasks:

(i) **Fault Management**: The effort to minimize the time required to detect and repair a problem in the network.

(ii) **Performance Management**: The effort to improve network performance and detect degrading performance by collecting and analyzing network statistics and modifying the appropriate network parameters.

The CSO operates a Proteon OverVIEW network monitor at its Network Operations Center (NOC). This monitor provides real-time information about disruptions in network service to an operator via a graphics display. It does not archive any of its data. It only indicates whether a router is down or a subnet is unreachable. Thus, it cannot indicate if a router is in trouble before it fails, nor can it generate reports over various time intervals. We needed a tool that could detect and isolate faults before they result in loss of service.

4.2. CENTRALIZED DATA COLLECTION AND MONITORING

It was decided to centralize data collection by using a single network monitoring station (NMS) operated by the CSO. Analysis of the data for routers belonging to the CSO is performed on this NMS too, but the major departmental network administrators perform the same analysis for their own departmental routers. At present, data concerning over 300 network interfaces is collected from about 100 managed entities (MEs). Each ME is polled hourly and almost one kilobyte of data is collected for each interface, making a 300 KB block of data for each sample interval.

Why not decentralize the monitoring function? We could divide the campus network into sub-domains, with one NMS for each sub-domain. Monitoring could then be done in a hierarchical fashion, with the NMS of each sub-domain reporting to a central campus NMS. One advantage of a decentralized implementation is that less traffic is generated because the MEs in the same sub-domain would report to their own NMS only a hop or two away, rather than a central NMS several hops away. Other advantages over a centralized system include greater reliability, and a smaller probability of the NMS being a bottleneck in the data collection process.

One problem of a decentralized implementation is that about 70 of the 100 or so MEs polled are backbone routers, which are considered "campus infrastructure", and come under the administrative domain of the CSO. The other MEs come under the administration of the various departmental networks or external networks. In order to realize the advantage of reduced traffic, the NMS would have to be on the departmental networks; this is administratively awkward for the CSO. In any case, as we will show later, our data indicates that the traffic generated by the monitoring function makes up a small proportion of the overall network traffic, so any gains in traffic reduction would be insignificant.

Another problem of a decentralized implementation is that it requires a NMS for each domain. At present, it is not economically justifiable for the CSO to invest in that many machines for the task of network monitoring. Even the present NMS, a Sun 3/50, is not dedicated to network monitoring alone.

How about designating one of the MEs, say one of the backbone redundancy routers, as a sub-domain's NMS? This poses a contractual problem because the backbone routers are all under a maintenance contract with the vendor. Installing monitoring software, or any other type of modification to these machines, may violate this contract. Even if this constraint did not exist, we would be biasing our data since the NMS's are also routers, and the monitoring data will be an added overhead for these machines. These routers will have to handle added traffic, with less CPU cycles and memory available for its routing function.

What if the NMS in a centralized implementation like ours fails? In order to avoid the total loss of monitoring capability, our monitoring software is duplicated on two other hosts. Since the current NMS is also used as a host, its failure can easily be detected with manual observation. Furthermore, the NMS is under constant watch by the OverVIEW monitor. It could then be arranged for the monitoring system to be activated on one of the alternative machines. A brief loss of monitoring data usually does not pose any serious degradation to our monitoring function.

4.3. POLLING vs ALERTS

There are two methods of transferring network management information from the ME to the NMS: request-response driven polling, and unsolicited sending of alerts [Ste89]. In a *polled* system, the NMS periodically sends each ME a request for network management information pertaining to that entity. On receipt of the request, the ME responds with the desired data. *Alerts* are management information delivered to the NMS that are not the result of a specific query, but because of the occurrence of some event (eg. receipt of a packet of data or occurrence of an output error).

The following factors have to be considered when choosing between these methods:

(i) **Reliable detection of failures**: The NMS should be informed of any failures that occur.

(ii) **Complexity and performance of the ME**: A management protocol that imposes less complexity on a ME will free more of its resources for its primary function.

(iii) **Response time for problem detection**: The time interval between the occurrence of a problem and its detection should be minimized.

(iv) **Volume of network management traffic**: The amount of network management information that is transferred between the ME and the NMS.

(v) **Ease of configuration**: It should be relatively easy for the NMS to configure the MEs.

Each method has its advantages and disadvantages. These are discussed in more detail in the following sections.

4.3.1. Polling

4.3.1.1. Advantages of Polling

(i) **Reliable detection of failures**: The NMS that polls the ME for management information can more readily determine failures in the ME or the network than an alert-based system. A lack of a response to a query indicates failures.
In the case of an alert-based system, the NMS assumes that the ME is good when it does not receive an alert, even when a failure has occurred and the alert message cannot be generated by the failed ME (eg. a power failure), or the alert message is generated but cannot reach the NMS (eg. a network partition).

(ii) **Reduced complexity of ME**: A polling system allows the NMS to initiate a poll when it is in a position to receive management traffic from an ME. There is no need to introduce any extra intelligence into the ME to handle the initiation of communications with the NMS, or any retransmissions.

Also, the use of polling allows the NMS to directly control the amount of management traffic across the network. There are no retransmissions unless the NMS asks for them. Thus, there is less chance that the NMS will be flooded with management packets at a time when it cannot handle them.

(iii) **Reduced performance impact on ME**: In a polled system, there is no need for the ME to generate an alert every time a monitored event occurs. It merely takes note of the monitored event whenever it occurs, and responds to a poll with the entire set of events that have occurred during the last sample interval. The performance impact of responding to a poll periodically is minimal, as long as the polling interval is not too frequent (eg. hourly compared to one minute intervals).

(iv) **Reduced requirements to manage a remote ME**: The remote ME need not be configured to report management information to specific destinations. The ME only responds to the requests it receives, so long as it is an authorized party (some authentication protocol is required here). Thus, the number of NMS's or the identity of the NMS which polls a particular ME can be changed without any need for reconfiguration of numerous MEs.

4.3.1.2. Disadvantages of Polling

(i) **Response time for problem detection**: The NMS may have to poll a large number of MEs, each for many parameters. It may be a while before the NMS can complete a cycle and poll a particular ME again. If an ME should fail immediately after it responds to a poll, there could be a significant delay before it is polled again. During this interval, the failed ME is assumed to be in good health.

A NMS could either use sequential or parallel polling. In *sequential polling*, the NMS waits for any outstanding requests to be responded to before polling the next ME. In *parallel polling*, the NMS can have a certain number of outstanding requests pending. Thus, for the same

number of MEs, the polling cycle for a particular ME can be smaller since the NMS spends less time in a busy waiting state.
In parallel polling, increasing the number of MEs results in more management traffic as well as reduced performance. In large networks, the NMS is limited by the number of polls that it can process in unit time. The larger network also requires the time-outs to be larger to accommodate responses that are still in transit. Also, a response could already have been received, but was queued while waiting to be processed. As a result, the time to detect a failure is increased.

(ii) **Volume of network management traffic**: Polling many parameters on many machines results in a large amount of network management traffic flowing across the network, and into the NMS to be stored. One way around this is to use hierarchical polling, where the NMS polls a set of intermediate stations for the general status of the machines that they poll. This localizes the management traffic, but results in increased time to failure detection.

4.3.2. Alerts

4.3.2.1. Advantages of Alerts

(i) **Real-time detection of problems**: Alerts allow the NMS to be notified immediately of a problem in the ME once it occurs. This eliminates the delay imposed by polling a large number of MEs, for a large number of parameters.

(ii) **Minimal amount of network management traffic**: Alerts are sent out only when errors are detected. By eliminating the need to transfer large amounts of "healthy" status information, network and system resources will be less heavily loaded.

4.3.2.2. Disadvantages of Alerts

(i) **Potential loss of critical information**: Alerts will not be delivered when the ME fails (eg. a power failure), or the network experiences some problems (eg. it is congested or partitioned). However, these are precisely the type of information that the NMS must receive. The lack of an alert will cause the NMS to assume that the network, or the ME, is in a healthy state.

(ii) **Potential to over-inform the NMS**: The NMS could be flooded by an excessive number of alerts in certain situations. For example, the status of a network link may alternate between "up" and "down", resulting in a series of "up"/"down" status alerts, or a ME may experience a long burst of errors, causing it to send an equally long burst of alerts to the NMS.
This flood of information places an extra burden on the network, as well as the NMS. It could even prevent the NMS from disabling the ME that is generating the alerts, because all available network bandwidth into the NMS, or the NMS itself, is saturated with incoming alerts.
In a polled system, the NMS need only be informed, in a single poll response, that the uptime for the vacillating link is less than the poll interval, or that a burst of errors was received.

(iii) **Increased performance impact on ME**: In an alert-based system, the ME must generate an alert every time a monitored event occurs. This involves forming the alert packet and sending it to the NMS. This places an overhead on the ME, depriving a possibly loaded processor of CPU cycles that could otherwise be used for its primary function.

(iv) **Increased complexity of ME**: An alert-based system requires that the ME alert the NMS whenever a monitored event occurs. This introduces an added complexity in the ME because the ME has to initiate alert messages to the NMS.

4.3.3. Reasons for Selecting Polling

We have decided to use a polling system because it was felt that reliable detection, minimal complexity and effect on performance of MEs, and ease of configuration of the MEs were more important, especially since there are a large number of MEs and only one NMS. Our strategy would be to put all the complexity into the NMS, and just let the MEs keep track of several simple performance variables needed to respond to a polling request. These variables are mainly counters (eg. the number of packets handled by the ME), or status flags (eg. whether an interface is activated or not).

Immediate notification of problems is not really necessary because there is enough redundancy in the network for minor failures to exist for the short term, while major failures can be reported by other means as well (eg. the OverVIEW monitor). Also, alerts might even lead to a large number of false alarms. The volume of management traffic generated by polling was judged to be relatively small compared to the normal amount of traffic on the campus network. Currently, each set of polls generates about 300 KB of data, and the polling interval is one hour. The campus network handles about 10 GB a day, so the polling data makes up about 0.1% of the total network traffic. Hence, this was judged to be less important compared to the potential of a flood of alerts saturating part of the network, or even the NMS, when an ME malfunctions.

An alternative would be to use a mixture of polling and alerts. Polling could be used to monitor variables that are likely to change frequently, while alerts are used to monitor more stable parameters. However, there is still the danger of the flooding problem associated with alerts.

5. DATA COLLECTION AND PROCESSING

5.1. SIMPLE NETWORK MANAGEMENT PROTOCOL

There are two non-proprietary network management protocols at present: the OSI based Common Management Information Services and Protocol Over TCP/IP (CMOT) specified in RFC-1095 [WaB89], and the Internet technology based Simple Network Management Protocol (SNMP) specified in RFC-1067 and RFC-1098 [CFS89]. While the Internet Advisory Board (IAB) has made CMOT an Internet standard, there is a lack of implementation of any tools based on this protocol. Thus, the only non-proprietary protocol currently in use on the Internet is SNMP.

5.1.1. SNMP Architecture

The Simple Network Management Protocol (SNMP), specified in the RFC-1098 [CFS89], defines a simple protocol by which management information for TCP/IP-based network elements, and, in particular, the Internet, may be monitored by a logically remote user. It is based on the design and architecture of its predecessor, the Simple Gateway Management Protocol (SGMP) [DCF88, DCF87].

The SNMP architectural model includes a collection of NMS's and network elements (which we have called MEs in the previous section). The NMS's monitor and control the network elements. The network elements are devices like hosts, routers, and terminal servers. Such devices have management *agents* responsible for performing the network management functions requested by the NMS.

The SNMP protocol is used to communicate management information between the NMS's and the agents in the network elements. It is primarily based on polling, although an alert-based facility is also available, but seldom used because of the potential loss of critical information and the potential to flood the NMS's with alert messages.

46

5.1.2. Goals of the SNMP Architecture

The first goal of SNMP is to minimize the number and complexity of management functions of the management agent. The benefits of this are:

(i) Reduced development cost for management agent software necessary to support the protocol.

(ii) Increased degree of management function that is remotely supported, resulting in the fewest possible restrictions on the form and sophistication of management tools.

(iii) Simplified sets of management functions are easily understood and used by developers of network management tools.

A second goal of the protocol is that the model for monitoring and control be sufficiently extensible to accommodate additional, possibly unanticipated aspects of network operation and management. A third goal is that the architecture be as independent of the architecture and mechanisms of particular hosts or routers as possible.

5.1.3. Representation of Management Information

Management information communicated by operation of the SNMP is represented according to the subset of the ASN.1 language [ISO87b] that is specified for the definition of non-aggregate types in the SMI. The SGMP adopted the convention of using a well-defined subset of the ASN.1 language. The SNMP extends this by utilizing a more complex subset of ASN.1 for describing managed objects and for describing the protocol data units used for managing those objects. In addition, the desire to ease eventual transition to OSI-based network management protocols led to the definition in the ASN.1 language of an Internet-standard Structure of Management Information (SMI) [RoM88] and Management Information Base (MIB) [McR88].

5.1.4. Operations on Management Information

The SNMP models all management agent functions as alterations or inspections of variables. Thus, a protocol entity on a logically remote host (possibly the network element itself) interacts with the management agent resident on the network element to retrieve or alter variables. This has two advantages:

(1) It limits the number of essential management functions of the management agent to two: value assignment to a variable, and retrieval of such a value.

(2) It avoids introducing into the protocol definition support for obligatory management commands (the number of such commands is in practice ever-increasing, and the semantics of such commands are generally complex).

The basic strategy of the SNMP is that the monitoring of network state is accomplished mainly by having the NMS's poll the network elements for the appropriate information. A limited number of unsolicited messages (traps) guide the timing and focus of the polling. Limiting the number of unsolicited messages ensures that the goals of simplicity, and minimizing the amount of traffic generated by the network management function, are not subverted.

5.1.5. Protocol Message Exchanges

The communication of management information between the NMS and the network elements is based on the exchange of protocol messages. To satisfy the goal of minimizing complexity of the management agent in the network element, the exchange of SNMP messages requires only an unreliable datagram service. Each message is entirely and independently represented by a single transport datagram. While the SNMP RFC-1098 [CFS89] specifies the exchange of messages via the UDP protocol, the mechanisms of the SNMP are generally suitable for use with a wide variety of transport services.

5.1.6. Management Information Base (MIB)

The Management Information Base (MIB) is specified in RFC-1066 [McR88, Ros89]. The MIB defines the variables needed for monitoring and controlling the various elements of the Internet. The list of variables used in this study can be found in Appendix A. Not all groups of variables are mandatory for all Internet elements. For example, the TCP group is mandatory for hosts running TCP but not for routers which do not run it. However, if any member of a group of variables is supported, then it is mandatory that all members of that group are supported.

The list of objects defined in the above-mentioned RFC are considered to be the essential management objects for either fault or configuration management. By considering only the essential objects, it was possible to limit the number to about 100 objects, making it easier for vendors to fully instrument these variables into their software. To avoid redundant variables, it was required that no object that can be derived from the other variables in the MIB be included in the MIB.

Although the number of objects defined in the MIB is small, additional private objects may be defined in the "enterprises" subtree of the MIB. These additional objects can be used for vendor-specific variables and for experimentation purposes.

5.2. DATA COLLECTION TOOLS

5.2.1. Querying Application: snmplog

The primary data collection tool for monitoring the UIUCnet is *snmplog*, a UIUC enhancement of the *snmpstat* and *snmppoll* tools provided by NYSERnet. Snmplog is a SNMP querying application for multiple agents. The agents are mainly routers, but could also be hosts, terminal servers, or any other machine that supports SNMP. It is designed to execute on a host and query any number of monitoring agents for information over an extended period of time.

Snmplog makes use of a configuration file to determine what information is requested, from which agent, and how often the information is requested. The configuration file also specifies various multipliers to a user configurable base querying interval, allowing the specification of the querying granularity at the level of both agents, variables, and sets of variables. Appendix B contains the Unix man page for snmplog in Section B.1, and a sample snmplog configuration file in Section B.2.

5.2.2. Types of Variables Monitored

The variables monitored can be divided into the Generic SNMP variables and the Enterprise (or vendor) Specific variables. Two groups of the Generic SNMP variables are monitored: the System Group and the Interface Table. Among the System Group variables monitored for each router are the system uptime and the system object identifier. The Interface Table variables monitored include the number of interfaces for each router, as well as the number of errors, discards, packets, and bytes into and out of each interface. Also monitored are the status and type of each interface.

Most of the enterprise specific variables monitored are Proteon variables because the vast majority of the routers in UIUCnet are Proteon routers. The other enterprise specific variables monitored are Cisco variables. Among the generic Proteon variables, which are common to all models of Proteon routers, the number of maintenance test and self test failures are monitored, as well as the type of network interface. Proteon routers carry out maintenance checks on themselves at regular intervals, and a pre-specified number of maintenance test failures are allowed before a more comprehensive self-test is performed.

Besides the generic Proteon variables, interface specific variables are also monitored for each type of network interface found on UIUCnet. These include ProNET-10 and ProNET-80 token rings; Interlan, Proteon, and Western Digital Ethernets; and 56 kbps and T1 serial lines. A detailed list of the monitored variables, and explanations of the meaning of each variable is given in Appendix A.

5.2.3. Agents Monitored

Data collection began on September 5, 1989, with the logging of only the Generic SNMP System and Interface Table variables (see Appendix A.1) of about 60 routers with almost 200 interfaces. At that time, the size of a typical log file was almost 2 MB, and this was reduced to a daily incremental file (called a delta file) of about 150 KB.

The number of routers monitored has since increased steadily. On December 11, 1989, vendor and interface specific variables (see Appendix A.2) were also logged and archived. The monitoring of about 70 routers at this time resulted in log files of almost 6 MB and delta files of about 350 KB. On January 22, 1990, the 11 backbone routers of CICnet, as well as the two MRnet routers on either side of the T1 line between MRnet's backbone and their router at UIUC, were added to the list of routers monitored.

As of April 1990, there are over 100 routers, hosts, and terminal servers, with over 300 interfaces being monitored. Approximately 80 of the routers are Proteon P4100s and P4200s, over a dozen are CISCO routers of CICnet and MRnet, and seven are NSC EN641s. Of the interfaces monitored, about 170 of them are Ethernet interfaces, 40 are ProNET-80 interfaces, 32 are ProNET-10 interfaces, 8 are HYPERchannel interfaces, over 50 are serial (T1 and 56 kbps) interfaces, and almost a dozen are ProNET-4 interfaces.

Considering the fact that there are such a large number of agents being monitored, and that most of them are more than one hop away from the NMS, how is monitoring affected by the failure of an intermediate router? Would all the routers downstream of that router be unreachable?

The redundancy of the campus network and its Internet connections comes to the rescue here. Failure of any of the routers that interconnect the campus' backbone rings will not affect the monitoring of the routers "behind" the failed router, because the backbone rings are redundantly interconnected. All three inter-connecting routers on a ProNET-10 ring or all four inter-connecting routers on a ProNET-80 ring must fail for it to be unreachable from the NMS [Fig 3.1].

Similarly, should any CICnet link be broken, traffic can take a longer path around either of the CICnet rings to their destination. Also, the CICnet has two connections to NSFnet, at UIUC and the University of Michigan at Ann Arbor. Should the CICnet get partitioned, or the CICnet router at UIUC fail, then the monitoring traffic could reach the NMS via the Ann Arbor site and the NSFnet.

5.3. DATA PROCESSING

5.3.2. Data Files

For the monitoring of UIUCnet, the queries are made hourly to all the agents specified in the snmplog configuration file. These agents are polled for traffic and error variables. In addition, the CICnet backbone routers are also monitored. They are polled at 15 minute intervals for traffic variables for the generation of link usage reports, and at hourly intervals for error variables. Daily summary reports of traffic and faults for all monitored agents, as well as a link usage report for CICnet, are produced each day. At the end of each day, a set of five data files would be obtained:

(i) A **log** file: This is the raw data file containing all the hourly data samples for all variables logged by snmplog in a given day. These are the values of the variable counters at the time they are sampled. These files have a ".log" extension to their file names.

(ii) A **delta** file: This contains the daily incremental data for all logged variables. This file is generated from the previous day's delta file (which contain the last counter values of all the variables in that day's log file) and the present day's log file. These files have a ".delta" extension.

(iii) An **account** file: This contains the daily incremental traffic data (in terms of packets and bytes) for each local routing domain; ie. the traffic into and out of each building or department, but excluding backbone and external interfaces. The data for the account file is obtained from the day's delta file. These files have a ".acc" extension.

(iv) A **sample** file: This contains the number of octets (rather than the value of the octet counter) over each sample of the CICnet backbone interfaces. This file is generated from the previous day's delta file (which contain the last counter values of all the variables in that day's log file) and the present day's log file. These files have a ".sample" extension.

(v) A **usage** file: This contains the link usage data (in terms of a percentage of the total link bandwidth) for each monitored link of the CICnet backbone interfaces. The information for the usage file is obtained from the day's sample file. These files have a ".usage" extension.

These files are referenced by the date, and archived in compressed form. Each record in these files represents the value of a particular variable, for a given agent's interface, at a particular point in time (in the case of log files) or over a particular time interval (for the rest). The data fields in these files are:

File	Field	Data Item
Log	1	Agent Name
	2	Interface Number
	3	Timestamp (seconds from 00:00:00 01Jan1970 GMT)
	4	ASN.1 object id
	5	SNMP variable value
Delta	1	Agent Name
	2	Interface Number
	3	Last Timestamp (seconds from 00:00:00 01Jan1970 GMT)
	4	ASN.1 object id
	5	Last SNMP variable value
	6	Timestamp interval (last timestamp - first timestamp)
	7	Incremental data over timestamp interval
	8	Counter indicating number of times data was lost
Account	1	Agent Name
	2	Interface Number
	3	Last Timestamp (seconds from 00:00:00 01Jan1970 GMT)
	4	ASN.1 object id
	5	Last SNMP variable value
	6	Timestamp interval (last timestamp - first timestamp)
	7	Incremental data over timestamp interval
	8	Counter indicating number of times data was lost
	9	Routing Domain name
	10	Net name
Sample	1	Agent Name
	2	Interface Number
	3	Current Timestamp (seconds from 00:00:00 01Jan1970 GMT)
	4	ASN.1 object id
	5	Current SNMP variable value
	6	Sample interval (previous timestamp - current timestamp)
	7	Incremental data over sample interval
	8	Counter indicating number of times data was lost
Usage	1	Agent Name
	2	Interface Number
	3	Highest Timestamp of a sample (seconds from 00:00:00 01Jan1970 GMT)
	4	Minimum Sample Interval of sample (seconds)
	5	Fractional Usage In
	6	Fractional Usage Out
	7	Spread of Sample Timestamp (seconds)
	8	Spread of Sample Interval (seconds)
	9	Counter indicating number of times data was lost
	10	Net name

A set of Unix C-shell and AWK scripts are used to process the data and generate reports. When the log file is being processed into a delta file, care must be taken to ensure that bad data is filtered out and not included in the delta file values. For example, a counter may wrap around or be reset, or the router may have restarted, at some point in time between polls.

To take such situations into account, the values stored in the delta files are always equal to, or less than, the actual number of events being counted. For example, suppose the last sample of the number of packets counter for a given interface returned a count of 4,000,000,000 packets, and at the current sample the counter returns a value of 8,000. It will be assumed that the interface being polled handled 8,000 packets during that sample interval, rather than (2^{32} - 4,000,000,000 + 8,000). Thus, the data recorded will always tend to be a conservative figure.

5.3.2. Summaries and Reports

At present, each hourly sample logs approximately 300 KB of data, making a 7.5 MB log file for each day. This is reduced to a 400 KB delta file and a 36KB account file for each day. In compressed form, they reduce to about a fifth of their original size, or about 1.7 MB in total. This is still a large amount of data for a day.

The data in the delta file is summarized into a daily report. This report is divided into the three sections shown below. In all three sections, the number of maintenance and self-test failures, errors, discarded packets, forwarded packets, and forwarded bytes are reported.

(i) **Data per Agent Interface**: In this section, the above-mentioned data for each interface of each agent that has a parameter over a specified threshold is reported. Also, if the router has been restarted (either by itself or by maintenance personnel), then this is reported too. Finally, an indication is given if some data for an interface is lost for some reason (eg. a bad sample point, counter wrap around, or router restart).

(ii) **Summary for each Network Interface Type for each Statistical Domain**: The above-mentioned data are collectively reported for each type of network interface, summing the data for all the interfaces monitored in a statistical domain.

(iii) **Summary for the Backbone Subnets and Major Departmental Backbones**: The above-mentioned data are reported on a subnet basis (rather than per agent interface) for the backbone rings and the backbone subnets of the NCSA, Beckman Institute, Department of Computer Science, and CSO's mainframes.

Monthly traffic reports are generated from the traffic accounting files. Since they are derived from the accounting files, they cover only the traffic information (ie. packets and bytes) for each local routing domain (which is usually a building or a department). Also, graphs for the above-mentioned traffic data are generated. Sample daily and monthly reports are given in Appendix B.

In addition to the traffic and fault reports, line usage data, based on the number of bytes handled by an interface, can be generated from the daily usage files. This data is generated for the CICnet backbone router interfaces. The daily usage reports provide information on the average line usage, as well as peak line usage and the hour at which the peak traffic occurred.

To help in the generation of various reports, a set of filters are available. These allow various types of data to be extracted from the various types of data files. It is possible to extract a given set of agents, a set of agent-interface pairs, a set of SNMP variables, or data in a given time interval.

6. FAULT DETECTION AND ISOLATION

6.1. OVERVIEW

Currently, fault detection and isolation are done mainly on a centralized basis. Data collection is done from a central NMS, seka.cso.uiuc.edu, a CSO Sun 3/50, and daily traffic and fault reports for the campus are generated for the campus network administrator.

The monitoring covers the backbone routers of the networks on the transit network *uir-uiuc-net*, and routers of UIRnet (ie. external connections); all the campus's backbone and exterior routers (ie. campus infrastructure); as well as certain internal departmental routers (ie. routers connected to departmental local networks) at the request of the departmental network administrators. These are mainly the major departmental networks, like the NCSA and Department of Computer Science. These departmental network administrators, as well as some network administrators of other networks (eg. MRnet), receive detailed fault reports for their own routers daily by electronic mail.

Fault detection is largely semi-automated now. It is mainly based on the reports being sent to the campus network administrator as well as the departmental network administrators. It is then up to these persons to analyze this data to determine which errors can be attributed to tangible faults, and which are spurious in nature.

Once it is decided that there is a problem which is causing the reported errors, the next step is fault isolation. Fault isolation is still basically a manual process based on more detailed analysis of the error parameters, usually over a period of time (with the help of archived data). Sometimes, experiments can also be carried out if these do not result in any effects on normal operational service; or, it might be possible to replicate those situations in the laboratory (non-disruptive testing). However, on other occasions, service must be interrupted in order to isolate a fault (disruptive testing).

6.2. FAULT DETECTION

6.2.1. Knowledge Base

For effective fault detection, a knowledge base of the typical fault characteristics of the campus network and its elements is required. This knowledge base is used to determine when a failure has occurred by using the typical fault characteristics to filter out random noise errors. It is important to ensure that the data in the knowledge base is accurate and up to date. Comparing the current network conditions with fault characteristics from some previous time may not produce the desired results (eg. older equipment could have been phased out leading to lower noise levels).

The fault detection knowledge base consists of per variable (and hence, per interface type) data. A set of data is kept for each known interface, as well as each statistical group of interfaces (eg. all the backbone interfaces). A 14 day history of all the SNMP variables is maintained in a file, called snmp.dev, and the mean and standard deviation is calculated for each variable.

The data used to generate the snmp.dev file is obtained from the incremental delta files. Only data from the 14 day window, ending on the previous day, is used. A more detailed discussion on the choice of the sampling window is given in Section 6.2.2. The data from the delta file 15 days ago is first subtracted from the snmp.dev file, and then the data previous day's delta file are added to it. By using a 14 day sliding window in this manner, it ensures that past errors that have been corrected, do not remain in the knowledge base indefinitely.

Specifically, the fields in snmp.dev are:

Field	Data Item
1	Interface or Statistical Group
2	Variable object ID (in ASN.1 format)
3	Sum of intervals, $\sum t$ (in seconds)
4	Sum of variable values, $\sum x$
5	Sum of square of variable values, $\sum x^2$
6	Mean (per second)
7	Standard deviation (per second)

The first two fields are character strings. The rest of the fields are in exponential format. The three sum fields are used to calculate the mean and standard deviation, with the data from each new day being added to these sums and outdated data being removed. Thus, they have an accuracy of 16 decimal places so as to minimize the cumulative error (which should be adequate for the 54 bit mantissa of a double precision floating point number used by AWK on a Sun 3/50).

The mean and standard deviation fields are used only for calculating a threshold. The threshold value is used for comparison against the values in the delta files, and not used in subsequent calculations of future values in snmp.dev. So, the problem of cumulative error is less significant with these two figures, and an accuracy of 10 decimal places is adequate (since the variable values are 32 bit integers).

The mean (per second) and standard deviation (per second) are then calculated as shown below.

$$\text{mean} = \frac{\sum x}{\sum t}$$

$$\text{std\_dev} = \sqrt{\frac{\sum t * \sum x^2 - (\sum x)^2}{(\sum t)^2}}$$

This formula for standard deviation facilitates the iterative calculation of the standard deviation values.

6.2.2. Factors Related to Sampling Window

The choice of the sampling window is affected by several factors. The *window size* could be *cumulative* or *fixed*. If it is a fixed size, then a suitable size must be chosen. A fixed window could be *static*, where it is determined once or irregularly. Or it could be *dynamic*, where it is recalculated each time new data becomes available.

6.2.2.1. Window Size

A cumulative window includes all samples from the beginning of data collection up to the current point. This will thus cover the entire history of the data. However, should any changes occur (eg. a router is upgraded, an interface is changed, or a different set of protocols used), the information would no longer be accurate for the new configuration. Also, any large bursts of errors will skew the data indefinitely since they will always be included in the mean and standard deviation values from the time of occurrence onwards.

When a fixed size window is used, choosing a suitable window size is important. In the case of dynamic windows, if the window is too small, then any error count that is increasing very slowly may not be detected because the mean value increases at a corresponding rate. If the window is too large, then there is a danger that the mean and standard deviation values may get skewed by error bursts, or abnormalities, for a long period of time. In the case of static windows, the window size should be large enough allow realistic average and standard deviation values to be obtained. In both cases, a very large window would approximate a cumulative window model.

6.2.2.2. Static vs Dynamic

For a static window, the history data is searched for a "typical" stretch of days which do not have any abnormally high or low error counts. The mean and standard deviation are calculated for this window. This need only be done once, or when there are significant changes to the situation which result in the window no longer being "typical".

The disadvantage of static windows is that the parameters are not always up to date with changing situations. Suppose a Proteon P4100 router needs to run DECnet, where a known bug in its software will often result in badly formed packets. These badly formed packets are reported as errors. After the initial detection of these errors, it may not be desirable for these errors to be reported again. There would then be a need to manually search for a new "typical" stretch of days that includes these errors. Use of a static scheme means that the network administrators must be on a constant lookout for such situations.

Using dynamic (or sliding) windows, the mean and standard deviation are automatically recalculated each time new sample data is received, or at regular intervals (eg. daily). Dynamic windows have the advantage of always being "current", and hence have greater accuracy in a changing environment.

However, dynamic windows have the danger of being skewed by abnormalities. For example, a burst of errors on a particular backbone ring over a period of a few days will raise the mean and standard deviation values while that period falls within the window.[1] The window size should be small enough to allow such a period to, pass through it in a reasonable amount of time.

6.2.2.3. Selection of a Fixed, Sliding Window Scheme

A fixed, sliding window scheme with a window size of 14 days was chosen in our system. The sliding window gave the advantage of always being current and could be calculated automatically. A 14 day window was chosen as it is felt that this large enough to give a reasonably accurate mean and standard deviation value, and yet small enough so as not to allow abnormalities to skew the data for too long a time.

6.2.3. Fault Reporting Thresholds

The daily reports, mentioned in Section 5.3.2, come in two basic forms. In the first form, it reports only traffic information and the generic SNMP error parameters. It does not report enterprise specific error variables. The second form provides a more detailed report. This includes all data reported in the previous form, plus the enterprise specific error variables that fall above the thresholds determined from the knowledge base. An example of this form of report is given in Appendix B.3.1.

---

[1] Another related example concerns Proteon Ethernet interfaces on Proteon P4200 routers. These interfaces seem to get very confused when ethernet cable taps are moved around while the router is still powered on. They develop a large number of errors, especially CRC and frame alignment errors. When the router is power cycled, the errors will go away, unless its cable taps get moved again. Soft resets do not seem to solve the problem.

The mean and standard deviation information in the snmp.dev file are used to generate the reporting thresholds. Fault parameters are considered statistically significant if they are above their respective thresholds. A threshold is calculated as follows:

threshold = mean + (n * std_dev)

where mean = mean number of a given type of error per second
over the past 14 days,
n = multiplier for the standard deviation,
std_dev = standard deviation per second over the past 14
days for the given fault parameter.

The resulting threshold is in terms of errors per second. A fault parameter is considered statistically significant if the following condition holds:

fault parameter > threshold * sample_interval

where sample_interval = sample interval over which the fault
parameter was measured (usually 24 hours).

By choosing an appropriate value for the standard deviation multiplier, n, it is possible to look at the interfaces that have the most significant number of errors. For example, for the 14 day periods shown below, the following percentage of routers with significant fault parameters were highlighted:

Multiplier	Top Percentage of Fault Parameters				Average
n	Jan1-Jan14	Jan15-Jan28	Jan29-Feb11	Feb12-Feb25	
0.0	48.3%	49.8%	56.5%	54.1%	52.2%
0.5	27.7%	29.9%	36.4%	36.8%	32.7%
1.0	23.0%	21.7%	28.9%	30.3%	26.0%
2.0	18.4%	16.8%	22.6%	23.1%	20.2%
3.0	15.7%	13.6%	18.5%	19.1%	16.7%

It was decided to set the threshold to the mean plus one standard deviation, so that we can roughly get the 25% of interfaces that are the worst offenders. By using the corresponding threshold to filter the data found in the delta files, we can incorporate the detailed error information for the worst 25% of the interfaces in the daily reports.

With the current number of agents monitored, this works out to about two dozen agents. It was felt that this was a reasonable number of agents for a human to monitor without being overwhelmed by data. It is believed that this threshold should allow us to catch all the agents that should be considered faulty, plus some amount of random noise. With more experience in the use of our tools, this threshold can be modified to minimize the level of noise.

The daily reports thus give the network administrator an indication of trouble on a given interface, router, or subnet, as well as how severe a problem is relative to the others. The next stage would be to isolate the problem and find its cause so that corrective action can be taken.

6.3. FAULT ISOLATION

6.3.1. Information Required

Fault isolation is basically the domain of the network management and maintenance personnel. The network monitoring and fault detection tools serve to complement the humans by organizing data so that they can look for commonalities. In general, the information required to help in isolating a fault can be divided into 4 basic categories:

(i) Router specific information:
- **Router name**
- **Vendor and model**
- **CPU type and memory size**
- **Software revision**

(ii) Infrastructure information:
- Interface type (eg. whether it is an ethernet or ProNET-10 interface, and, if ethernet, whether it is an Interlan, Proteon, or Western Digital interface)
- Subnetwork number
- Network topology (eg. position in network and wire center for backbone rings)
- Protocols supported (eg. IP, DECnet)

(iii) SNMP variables:
- Number and frequency of restarts
- Number of errors and discards
- Number of packets and bytes handled
- Number of maintenance and self-test failures
- Interface specific variable counts for that interface type

(iv) Others:
- Any relevant information about its past history (eg. a router may have been a beta-test prototype)

6.3.2. Examples of Successful Fault Isolation

In this section, some examples of successful fault isolation are given. In these examples, the faults are detected with the help of the daily reports, isolated to a particular network element (hardware or software), and corrective action was successfully taken to eliminate the fault.

The first example deals with the continuous restarting of a router. Normally, a router will restart itself when it discovers some inconsistency in its internal state. For example, a router may have failed a given number of self-tests, or there were fluctuations in the power supply.[2]

The second example concerns the high number of badly formatted packets found on a particular backbone ring. Although the number of errors was high, it was not high enough to affect all users on that ring. It involved packets of a particular type of protocol, and so only users of that protocol noticed some effect on network service.

6.3.2.1. Continuous Restarting of enghall.gw Router

Fault Detection: In the second week of January 1990, it was noticed from the daily fault reports that the router enghall.gw.uiuc.edu was restarting itself almost every day, and often several times a day. A hourly plot of the System Uptime counter of that router confirmed that the router was being restarted at irregular and frequent intervals. This can be seen in Fig 6.1, where the System Uptime count was reset to zero 45 times over the 21 day period from December 27, 1989 to January 16, 1990 (Hour 0 to 493). The 32 bit System Uptime counter normally takes about 500 days to wrap around.

---

[2] In February 14, 1990, the campus was hit by an ice storm which resulted in major power fluctuations when power lines went down in the county area. Twenty-eight of the 100 routers monitored restarted themselves between one to three times that day. However, the network service was not seriously affected and the number of errors reported by the routers were not significantly above average.

Fault Isolation: The lack of any similar fault patterns in neighboring routers indicated that the problem was local to the enghall.gw router. This particular router is a Proteon P4200, which consists of a CPU board and the various interface boards, all plugged into a common backplane. The various interfaces on the router showed no significant abnormalities. It was suspected that the CPU board in that router was at fault.

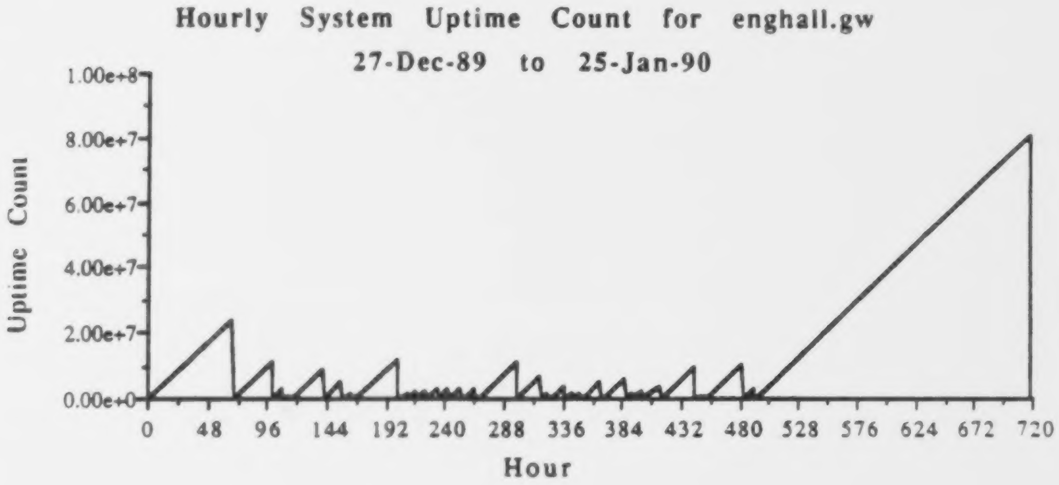


Fig 6.1: System Uptime count of enghall.gw

Fault Correction: At about 1:30 pm on January 16 (Hour 493), the CPU board on this router was replaced. After that, the problem no longer manifested itself, with the System Uptime counter no longer returning to zero over a period of 9 days, ie. January 16 to January 25 (Hour 493 to 717), as shown in Fig 6.1. This indicates that the CPU board was the culprit and that the problem was successfully corrected.

6.3.2.2. Badly Formatted Packets on P4100 Routers

Fault Detection: In November 1989, it was noticed that there were high Input Error counts on several routers on the 128.174.7 ProNET-10 backbone ring. Looking at the data collected, it was discovered that the Proteon P4100 routers connected to the same wire center in Node 2 (one of the five nodes on campus where the wire centers of the backbone rings are located) were the most seriously affected by this problem.

Fault Isolation: Closer scrutiny of the error data revealed that six of the seven routers on that wire center had a higher number of Input Errors than Output Errors. One router, named ps.gw.uiuc.edu, had input and output errors that were of the same order of magnitude. Most of the errors were due to badly formed packets. It was concluded that the fault was either a problem with the network hardware between the routers upstream and downstream of ps.gw [Fig 6.2], or it could be a hardware or software problem in ps.gw itself.

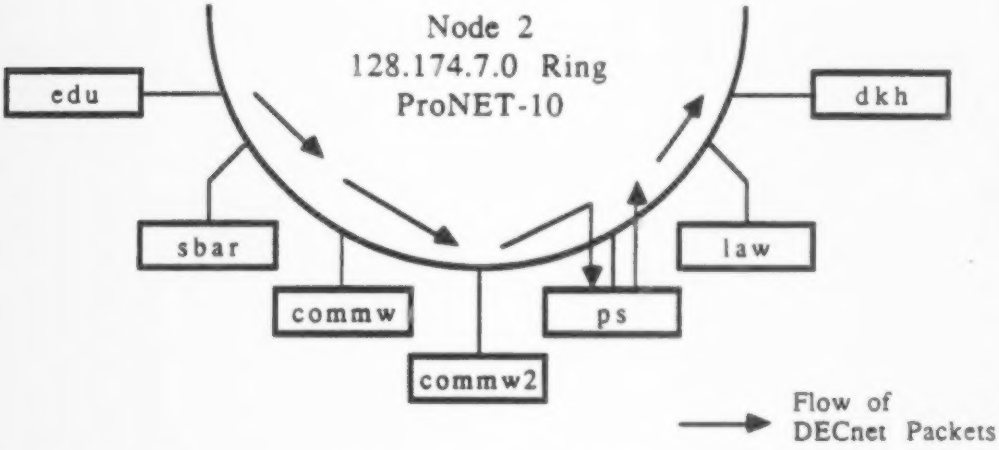


Fig 6.2: Routers connected to the 128.174.7 Ring wire center in Node 2.

While trouble-shooting for the fault, several minor problems, like improperly tuned modems (which convert optical signals into electrical signals and vice-versa), were discovered. The correction of these contributed to minor reductions in the error counts. A chronological log of the events can be found in Appendix C.

In late November 1989, DECnet users attached to the local network side of ps.gw complained about poor DECnet service. It was realized that the problem could be due to the implementation of DECnet support in Proteon's Version 8.1a software, because ps.gw was the only router on that wire center running DECnet. Deactivating DECnet on ps.gw resulted in a significant decrease in errors on the ring, thus confirming the suspected fault.

This bug in software for P4100 routers explains the pattern of errors. The router ps.gw was both receiving and generating DECnet packets, hence it had input and output errors of the same order of magnitude. The routers downstream of ps.gw had high Input Error counts since they were receiving badly formatted packets generated by ps.gw. The interfaces of the routers upstream of ps.gw were receiving and regenerating DECnet packets returning to ps.gw after making one round of the token ring, and thus also experienced badly formatted input packets.

Fault Correction: In early December 1989, the P4100 router ps.gw was temporarily replaced with a Proteon P4200 router running Version 8.1 software, since similar routers on the same ring were not affected by the problem. The problem was found to be a bug in the Version 8.1a software that ran on the Proteon P4100 routers.

The DECnet Broadcast Routing Timer had a default of 180 seconds. With this value, there were no problems with badly formatted packets. However, in late October 1989, the value for this timer was set to 600 seconds. This resulted in badly formatted packets. It was found that any positive value other than the default returned a negative number when queried. In mid-December 1989, ps.gw was again switched back to a P4100 with its Broadcast Routing Timer set to the default value, until a new version of software was available for the P4100s.

6.3.3. Examples of Less Successful Fault Isolation

In this section, some examples of less successful fault isolation are given. In these examples, the faults are detected and isolated as before. However, it was not possible to isolate the fault to a particular problem, so it was not possible to take definite corrective action.

The first example deals with the recurring restarts of a router. The second example concerns the high number of discarded packets on a particular router's interface. The third example shows how a malfunctioning host in one administrative domain can affect another administrative domain.

6.3.3.1. Recurring Restarts of pintail.gw Router

Fault Detection: During the period October 27, 1989 (Hour 0) to December 20, 1989 (Hour 1336), it was noticed that pintail.gw.uiuc.edu was also restarting itself quite frequently. The hourly plot of its System Uptime counter [Fig 6.3] shows that it restarted 48 times during that interval.

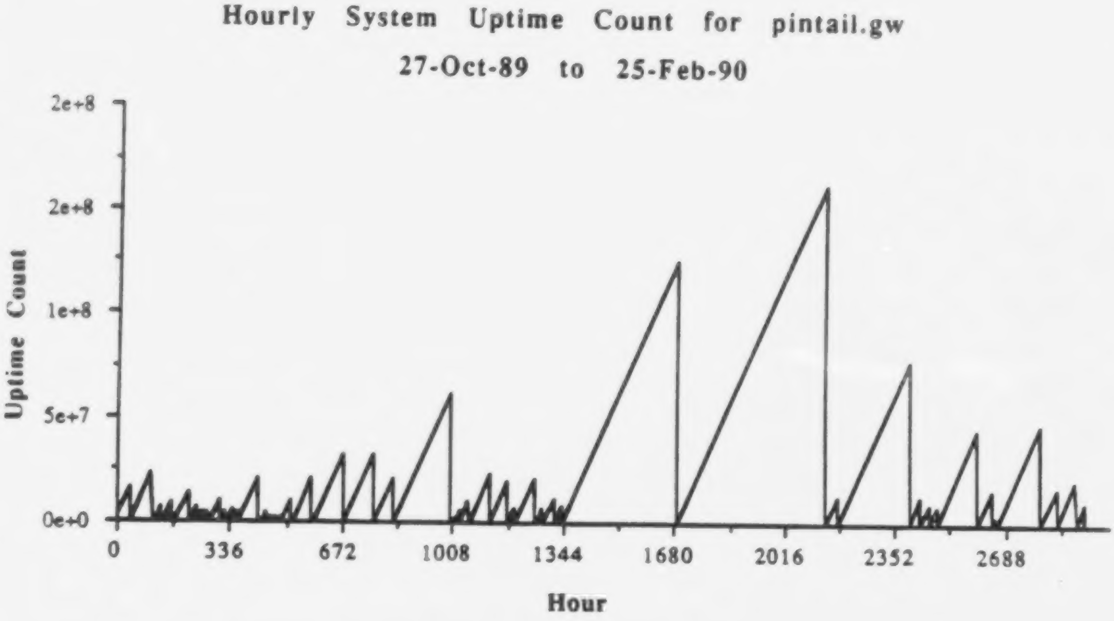


Fig 6.3: System Uptime count of pintail.gw

Fault Isolation Attempt: As the various interfaces of the router did not show any abnormalities, it was suspected that the system software or CPU board was at fault. It was decided to do a hard reset by powering down the router and then powering it up again. This was done on December 20 (Hour

1336). After power cycling the router, it became stable again, as can be seen over the period December 20, 1989 (Hour 1336) to January 23, 1990 (Hour 2141), with the exception of a restart on January 4, 1990 (Hour 1686).

It is suspected that the problem is due to some bug in the system software or hardware counters/registers that begins to manifest itself after a few weeks, and can only be reset by power cycling the router. Anything short of a power cycle will not reset an EN641 router to its initial state.

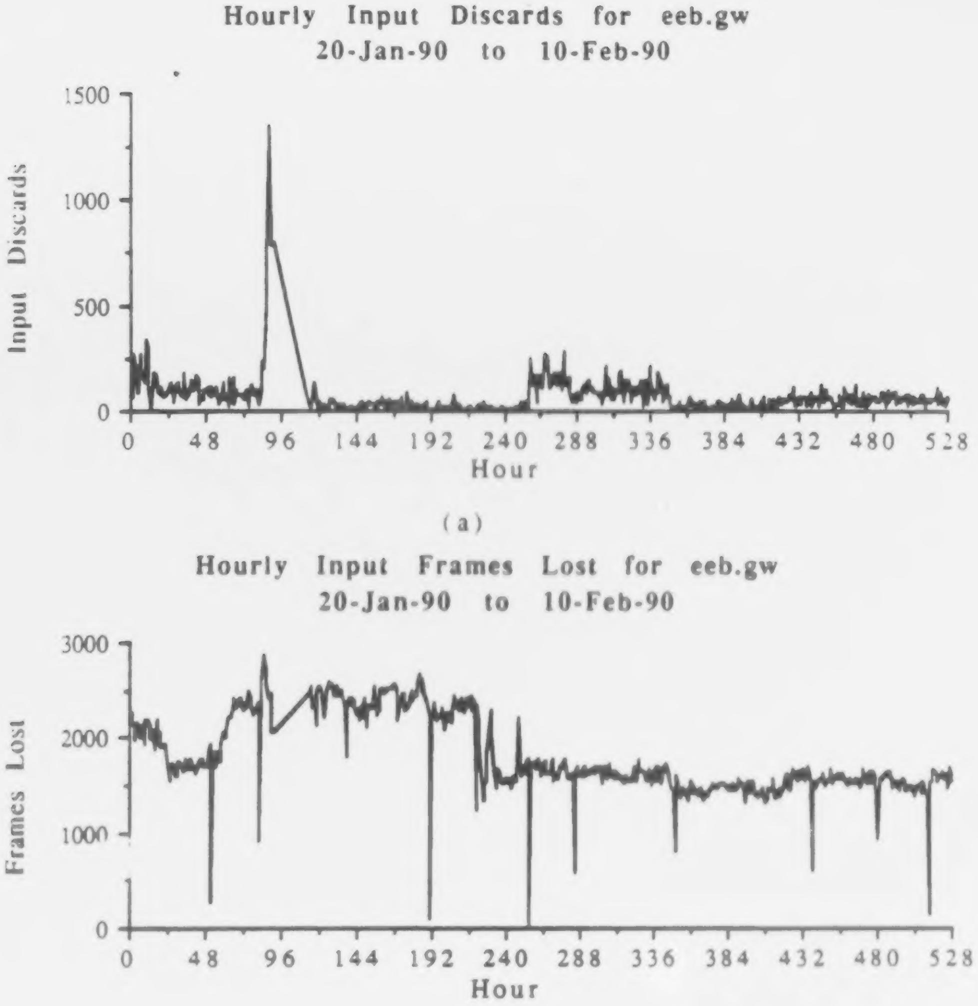
6.3.3.2. Input Discards and Frames Lost on eeb.gw Router

Fault Detection: In late January 1990, it was noticed that eeb.gw.uiuc.edu, a Proteon P4200 router, was experiencing high error counts on one its Proteon ethernet interfaces. These were mainly input discards and frames lost errors. Plots of the Input Discards and Frames Lost counts on that interface can be seen in Fig 6.4.

Fault Isolation Attempt: Usually, an interface generates Input Discards and Frames Lost errors when it has insufficient buffer space to cope with all the arriving packets (See Appendix A for the full definition of these variables). From the hourly plots in Fig 6.4, it can be seen that these errors occur at a fairly steady rate.

However, for such errors to occur, it would be necessary for the router to receive a flood of incoming packets at a rate that is faster than the router's ability to forward the packets. This seems to indicate that there is a node on the router's local network that is sending bursts of packets to the router fairly frequently, about once an hour, perhaps creating miniature broadcast storms. Another possibility is that there is a fault on that interface, causing it to incorrectly discard, or report the discarding of, packets.

Since the problem is mainly on the local network side (rather than the backbone side) of the router, it falls within the domain of the local network administrator. Data concerning the problem can be forwarded to him, and any action that needs to be taken will be at his discretion.


Hourly Input Discards for eeb.gw
20-Jan-90 to 10-Feb-90

(a)

Hourly Input Frames Lost for eeb.gw
20-Jan-90 to 10-Feb-90

(b)

Fig 6.4: Input Discards and Frames Lost on eeb.gw

6.3.3.3. Domain Loop from Indiana

Fault Detection: It was noticed from the Daily Line Usage reports for CICnet that the average line usage figures on February 25, 1990, for the T1 link between the CICnet backbone routers of Indiana University (IU) and UIUC (named iu.cic.net and uiuc.cic.net respectively) was abnormally high. Extracts of IU's line usage figures for February 25, and, for comparison, February 22, can be found in Appendix C.2.1

On February 25, a Sunday, the average line utilization was about 10% while the peak value was about 12%; on Thursday February 22, the corresponding figures were about 2% and 16% respectively, which is fairly typical for a weekday. This indicates that there was a consistently high line utilization over the entire day February 25. Checking the raw data collected for that day confirmed this assumption.

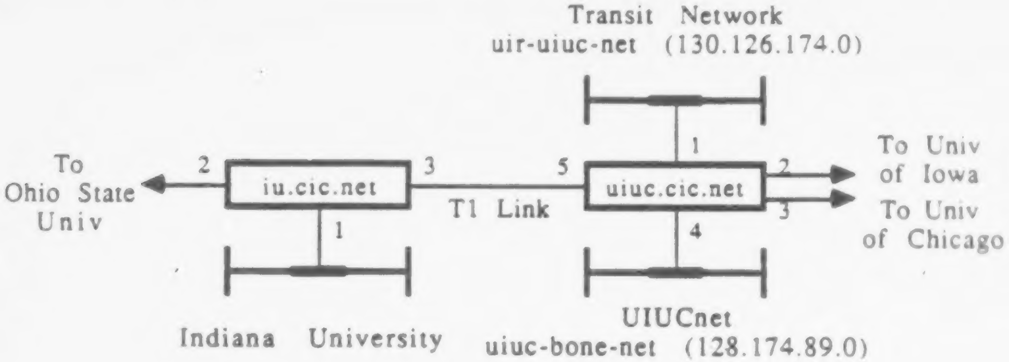

Transit Network
uir-uiuc-net (130.126.174.0)
To Ohio State Univ
2
iu.cic.net
3
1
T1 Link
5
uiuc.cic.net
1
2
3
4
To Univ of Iowa
To Univ of Chicago
Indiana University
UIUCnet
uiuc-bone-net (128.174.89.0)

Fig 6.5: CICnet link between IU and UIUC

Fault Isolation Attempt: From the Daily Fault and Traffic reports for those two days, it was possible to deduce that a significant portion of this traffic flowed onto the transit network, ***uir-uiuc-net*** (130.126.174.0). Extracts of the Daily Fault and Traffic reports for February 22 and February 25 can be found in Appendix C.2.2.

Looking at the number of unicast packets (labeled "UcastKpkt" in the reports) handled by the router iu.cic.net on February 25, it can be seen that about 5 to 6 million packets flowed between IU and UIUC. On the same day, the number of unicast packets handled by uiuc.cic.net's interfaces with the University of Iowa and University of Chicago were between 1 to 3 million. This figure is fairly typical. However, the traffic flow to the router's interfaces with the transit network *uir-uiuc-net* (130.126.174.0) and UIUCnet's *uiuc-bone-net* (128.174.89.0) was above average.

A network monitor, running MIT's PC/IP Netwatch program, is attached to the transit network *uir-uiuc-net*. This monitor allows us to look at each packet flowing on that network and determine its protocol, packet type, and source

A network monitor, running MIT's PC/IP Netwatch program, is attached to the transit network *uir-uiuc-net*. This monitor allows us to look at each packet flowing on that network and determine its protocol, packet type, and source and destination IP address. It was found that there was a steady and significant amount of UDP Domain traffic, with almost every Domain packet on the transit network involving a particular IU host as either its source or destination. It was concluded that that host was in a Domain loop, causing it to send out Domain packets continuously.

Since the errant host fell within the domain of the network administrator for Indiana University, data concerning the problem was sent to him by electronic mail, and it was left to him to take action to rectify the problem.

7. EVALUATION AND COMMENTS

7.1. MONITORING WINDOW

At present, the primary tool for monitoring fault and traffic data are the daily reports, which are ,based on hourly polls of the monitored network elements. Since the size of the monitoring window for these reports is only one day, there is a risk that they may not detect "creeping" problems (ie. faults that manifest themselves at a slow but steadily increasing rate), or longer term trends (say, in traffic patterns), which would only be noticed using a larger monitoring window. Although, adding reports with larger monitoring windows (eg. weekly or fortnightly) will help detect failure trends, it presents even more data for analysis.

Also, there is already a large quantity of data generated by the daily reports covering over a hundred network elements. Currently, the analysis of this data is still largely manual, although there are filters based on the mean and standard deviation of the error parameters to allow the worst 25% of the erroneous network elements to be highlighted (see Section 6.2). There is still a need to decide which of these two to three dozen routers to pay special attention to each day, as well as to determine how many of these still fall into the random noise category.

7.2. DATA COLLECTION AND PROCESSING

At present, the data collection is run as a background process on a Sun 3/50 workstation with 8 MB of memory, and it takes an average of 4 minutes to poll all the network interfaces of over 100 agents. However, this workstation is not dedicated to just network monitoring, but it also takes part in the EGP protocol reachability announcements on the transit network *uir-uiuc-net*, as well as being used as a host. When this workstation is heavily loaded, a set of polls could take as long as 25 minutes.

So, for the number of interfaces and variables monitored now, and the present polling interval, it is still well within the workstation's capability to finish polling all of agents before the start of the next polling period, an hour later. If the number of agents is doubled or the polling interval halved, it would still be within the workstation's capability under normal loads, although there may be some problems when the workstation is heavily loaded. Also, the size of the log files will be doubled.

Data processing of the log files into the four other types of data files is done after midnight each day. It usually takes about half an hour to process all the data on the Sun 3/50. Since it is done after midnight, idle CPU cycles are available, and no user is affected by the data processing run.

Currently, snmplog outputs the raw data to a single log file. The size of the log file is now getting to be too large for efficient retrieval of specific archived data. It might be better to have separate log files for different data, like traffic data, error data, status information and so on. Thus, an improvement to the snmplog routine would be to be able to output to multiple log files; eg. we could have different log files for different sets of variables or groups of agents.

7.3. DATA ANALYSIS

At present, analysis of the data is done manually now, mainly by the campus network administrator. With present quantity of data, it is just within the capability of the human network administrator to detect most of the faults as confirmed by our experience. Even with the departmental network administrators analyzing the data concerning their own local networks, there are still about 70 backbone routers, as well as the routers on the transit network ***uir-uiuc-net***, that are considered "campus infrastructure", and thus fall under the domain of the campus network administrator.

Thus, although the tools available at present are adequate for the monitoring of the smaller departmental network infrastructure, there is a need for more intelligent tools to help in the monitoring of the campus infrastructure. Artificially intelligent (AI) tools, or expert systems, are probably necessary to perform this task. These tools must be able to look at the parameters of the various agents and interfaces monitored over various time frames (eg. daily, weekly, and monthly), and highlight these for corrective action when necessary.

A rule-based approach could be used with a simple set of if-then rules for fault detection, and a more stringent set for fault isolation. The fault detection rules could simply be based on the number of times an agent had errors over its particular threshold in a particular monitoring window. In the example on the continuous restarting of the enghall.gw router (Section 6.3.2.1), a rule restricting the number of restarts of an agent in a seven day period could have detected the problem.

The fault isolation rules would need to pay more attention to detail. They must pay attention not only to the agent, but also to the suspected interface, and the number of each type of error reported. Consider the example concerning the DECnet problem (Section 6.3.2.2). The tool must be able to list out all the common factors among the erroneous agents: router type, software version, interface type, and high input error counts. It should then look at what was irregular: the fact that only ps.gw had roughly the same number of input and output errors, and that it was the only one with DECnet support enabled.

A more intelligent tool may even prioritize the faulty agents or interfaces according to some rule base defined by the network administrator. Ultimately, the tool should be able to recommend tests, whether disruptive or non-disruptive, which the network operators can perform. The results of these tests can be used to isolate faults with greater accuracy.

To build up the rule-base, a fault model of the network needs to be built up. The fault model would consist of a list of faults, and the symptoms of each. Experience from the current monitoring tool can be used to help build up this fault model. With the fault model, the rule-base can be built up in the form of a mapping from a symptom to all the faults that can cause this symptom.

For this AI tool to be effective, it must have a readily accessible database of information. Currently, there are separate databases for the software configurations, network topologies, router parameters, and fault thresholds. Integrating all this information into a common, easily accessible database would facilitate its use by the AI tool.

7.4. ADMINISTRATIVE STRUCTURE

In Section 6.3, we have seen successful and less successful examples of fault isolation. For the successful examples, one of the reasons for the success was that the cause as well as manifestation of the fault fell within the same administrative domain. For instance, the DECnet problem on the ProNET-10 ring 128.174.7.0 (Section 6.3.2.2) had the fault manifested as high input error counts on the routers connected to a particular wire center. The cause of the problem was isolated to a backbone router attached to that wire center, which falls within the same administrative domain as the backbone ring and wire center.

For the less successful examples, one of the reasons why the exact cause of the problem was not found is that the cause of the fault was in a separate administrative domain. It would then be up to the network administrator of that domain to take action to isolate and correct the faults. For instance, in the problem with the CICnet Domain loop (Section 6.3.3.3), the manifestation of the fault was a high line usage between the CICnet backbone routers at Indiana University (IU) and UIUC. The cause was an errant host in IU, which, of course, falls outside the domain of UIUC'c campus network administrator.

Therefore, the autonomy of the various administrative domains imposes a constraint on the technical solution to some of the problems detected, making it not always possible for totally centralized fault detection, isolation, and correction. When a fault in some department's administrative domain affects a network element in the campus infrastructure, the campus network administrator is dependent on the departmental network administrator to act on correcting the fault. However, the departmental network administrator is often dependent on data from the campus network administrator to isolate the fault.

This highlights the need for greater cooperation between the network administrators of the various administrative domains: departmental, campus, as well as those of other campuses or organizations. Also a clear demarcation of areas of responsibility is essential.

7.5. FAULT DISTRIBUTION

In Bernstein and Yuhas's paper [BeY89], they give the following breakdown of faults in telecommunications networks: 40% of the faults are due to hardware and software provided by the vendor, 25% are due to application software, 15% to communication line problems, 10% to environmental problems, and the remaining 10% to operational problems.

At present, we do not have sufficient data to give such a breakdown for UIUCnet. However, it is believed that, in general, we do have a similar fault distribution pattern. This would support our emphasis on trend analysis of the various network elements in terms of traffic and fault parameters, and communication lines in terms of link usage.

7.6. EXTENSIONS REQUIRED FOR HOST MONITORING

What if it is desired to monitor all machines (hosts, terminal servers, routers, etc) attached to the campus network? Then decentralization of the monitoring function into some hierarchical structure is essential. Each physical network will probably need its own NMS. The NMS can then filter out information on the worst offenders (eg. by taking the mean number of errors plus one standard deviation as a threshold) to be forwarded to a departmental NMS, and so on up the hierarchy. Thus, going down the hierarchy, we get an increasing degree of detail in the fault parameters monitored; while going up the hierarchy, we get summaries over larger monitoring windows and a statistically more significant set of offenders.

Another point to consider is that support of the current version of the monitoring protocol, SNMP, among hosts is much poorer than among routers.

Often, hosts do not even support the complete set of basic Generic SNMP variables. Also, SNMP was initially targeted mainly at routers rather than hosts. It is currently being expanded to be more general to take care of its current inadequacies. Alternatively, another protocol specifically targeted at host monitoring could be developed. This protocol could include a simplified subset of the SNMP variables, since hosts will not need to support certain router protocols, like the Exterior Gateway Protocol (EGP) [Com88], which allows exterior routers to exchange routing information.

Another factor to consider is the cost of monitoring in terms of traffic overhead generated by the polling packets, the number of NMS's required, and the personnel costs to develop and maintain the monitoring tools as well as to perform the monitoring functions.

7.7. RELATED WORK

7.7.1. Network Monitoring Tools

In Soha's work on distributed LAN monitoring [Soh87], he uses intelligent high performance monitors (IHPMs), based on data link bridges, together with a few management nodes (MNs), which are general purpose computers. His monitors are "intelligent" because they do not require a specification of what the monitor should examine. The IHPM simply receives every packet, and learns the required information to provide meaningful statistics for the LAN. The IHPMs are not general purpose but are of different types, each type optimized for a particular function. They are also dual-ported (i.e., they have two LAN interfaces) allowing them perform real-time monitoring by having one interface connected to the LAN to be monitored, and another interface connected to a LAN for reporting purposes. This feature also allows the IHPM to monitor heavily loaded LANs unobtrusively, or simply serve as a redundant port should the other one fail.

The MNs are used to perform data reduction on the data received from the IHPMs. They are capable of providing real-time statistics, as well as archived data for trend analysis. Although the MNs get their data from the IHPMs via

77

unsolicited messages, a user can request specific data via solicited messages. A connectionless datagram communication protocol is used between the MNs and the IHPMs.

Feridun et. al. describe the Automated Network Management System (ANM) [FLN88] which provides an integrated set of tools for real-time monitoring, control, and analysis of internets consisting of diverse network entities like routers, packet switching nodes, and hosts. Its architecture consists of a backbone of Distributed Management Modules (DMMs) and clients. The DMM backbone is responsible for gathering raw data and controlling the network entities. Clients request and receive raw data collected from the network entities by the DMM backbone, and can also request the DMM backbone to execute specific control actions. The clients analyze, archive, and present the gathered data to the network operator. A Network Management Protocol (NMP) provides a standard interface between the clients and the DMM backbone. A similar architecture can be found in Cassell and Amer's distributed measurement system [CaA88].

Our work is similar to the work discussed above in that we separate the data collection function from the data processing function; in our case, the MEs maintain a set of variables as defined by SNMP, and the NMS collects the data periodically from all the MEs for processing and reduction. Also, only one NMS is used, with a larger number of MEs.

However, our MEs can be any device (router, terminal server, or host) that supports SNMP; the IHPMs and DMMs tend to be special purpose devices. With dual-ported IHPMs, Soha places a great deal of flexibility into the monitors, while our approach tends to be more economical. We have no special purpose hardware and we have placed most of our complexity in our NMS, leaving our MEs as simple as possible. Communication is thus by polling, rather than unsolicited messages (or alerts).

The management stations are based on general purpose hosts, and they all perform data reduction and report generation. Communication between the management stations and monitors is based on some connectionless datagram protocol, in our case UDP, for simplicity. However, our NMS does not include a real-time monitoring function, since we already have such a dedicated monitor. Our emphasis is on trend analysis for fault detection and traffic monitoring.

7.7.2. Artificially Intelligent Tools

Most of the work on artificially intelligent tools and expert systems for network monitoring have been in the area of data or telecommunications networks, rather than computer networks.

Sutter and Zeldin [SuZ88] describe an expert system for real-time diagnosis with goals and a network model that are similar to ours. Their diagnostic system should be able to present its diagnosis and repair recommendations before the network operator becomes aware of the problem. Their network tends to be highly redundant at the component level, and highly interconnected at the network topology level. Thus, they must often diagnose faults that have not yet been noticed by the network users. Their system has to maintain an accurate model of the network configuration, reason with incomplete data, track changing problems over time, sort out root problems from side-effects, order problems in terms of severity, and select the cheapest and most informative tests available.

Marques [Mar88] describes the *Starkeeper* Network Troubleshooter which is an interactive symptom driven expert system for isolating and correcting virtual circuit network faults. Based on the given symptoms and background information, the *Starkeeper* identifies all suspicious components, formulates a plan to minimize the time taken to isolate the fault, executes the plan, and terminates when the fault is isolated and corrected. The *Starkeeper*'s outputs are: (i) a diagnosis identifying the physical address of the faulty network element, (ii) results of any system initiated procedures to clear the fault, and (iii) advice on appropriate corrective procedures if human intervention is required.

We expect our knowledge base to be similar to that of the Diagnostic Repair Expert System Shell (DRESS) [RBS88]. DRESS contains the following domain knowledge entities: symptoms, actions and recommendations, heuristic rules, detailed repair procedures, static domain knowledge, and site-specific domain knowledge.

In addition to the above, an introduction to rule-based expert systems for network management and operations can be found in [CCB88], and a survey of expert systems for fault detection, test generation, and maintenance can be found in [Pau86].

8. CONCLUSION

In summary, we have shown the need for a monitoring tool to help in the management of a large campus computer network. We have also discussed the reasons for choosing a centralized polling system, and the choice of the SNMP protocol. Analysis of the monitored data is distributed among the campus network administrator and the administrators of the major departmental networks.

At present, the fault detection process is semi-automated with the help of the monitoring tool. However, the fault isolation process is still largely manual. It is hoped that with more experience in monitoring the network, and with the help of artificially intelligent tools, it will be possible to increase the degree of automation.

Our monitoring tool can be characterized as follows:

(i) **Separation of Function**: The managed entities MEs) maintain cumulative counters and status flags internally, and operate autonomously. Periodically, these MEs are polled by a network management station (NMS), which then processes the data collected daily into a more manageable amount for reporting purposes.

(ii) **Economical**: A single NMS is able to monitor a large number of MEs. No modification of the MEs (other than enabling support for SNMP), or special hardware or software is required for the MEs.

(iii) **Minimum Complexity or Performance Impact on MEs**: The system complexity is mainly in the software tools running on the NMS. The MEs simply increment their counters whenever the relevant events occur, and respond with the values of these counters whenever they are polled by the NMS. This also minimizes the performance impact on the MEs.

(iv) **Flexible Coverage**: It can be used to monitor agents on a LAN, MAN, or WAN. All it requires is that the agent supports the SNMP protocol, and its defined Generic Variables.

A. SNMP VARIABLES MONITORED

All variables should be reported as 32-bit values, although in some implementations, they are stored as 16-bit values.

A.1. GENERIC SNMP VARIABLES

The Generic SNMP variables are defined in the Management Information Base (MIB) for Network Management of TCP/IP-based internets, RFC-1066 [McR88]. Work is in progress on a draft specification of the newer MIB 2 [Ros89].

A.1.1. System Variables

The System variables are mandatory for all systems that support SNMP. These variables are maintained on a "per box" basis.

sysObjectID: The vendor's authoritative identification of the network management subsystem contained in the managed entity. It provides an easy and unambiguous means for determining "what kind of box" is being managed.

sysUpTime: Time (in 0.01 sec) since network management portion of system was re-initialised. This counter will wrap around in approximately 500 days.

ifNumber: Number of network interfaces (regardless of their current state) on which the system can send and receive IP datagrams.

A.1.2. Interface Table Variables

The Interface Table variables are mandatory for all systems that support SNMP. These variables are maintained on a "per interface" basis.

ifType: Interface type based on the Physical, Data Link and Network Layer protocols "below IP". These include:

- 1822
- IEEE 802.3/4/5/6
- ProNET-10/80
- FDDI
- SDLC
- ISDN
- X.25
- StarLAN
- HYPERchannel
- LAPB
- T1/CEPT
- Point-to-point Serial

ifAdminStatus: The desired state of the interface.

ifOperStatus: The current operational state of the interface.

ifInOctets: Total number of octets received on the interfaces including framing characters.

ifInUcastPkts: Number of subnet unicast packets delivered to a higher layer protocol.

ifInNUcastPkts: Number of subnet non-unicast (ie. broadcast or multicast) packets delivered to a higher layer protocol. This variable is not supported by all agents. In agents where it is not supported, non-unicast packets are counted together with unicast packets.

ifInDiscards: Number of inbound packets which were chosen to be discarded even though no errors had been detected. Eg. to free up buffer space when the input buffers are full.

ifInErrors: Number of inbound packets that contained errors that prevent them from being delivered to a higher layer protocol.

ifOutOctets: Total number of octets transmitted out of the interface including framing characters.

ifOutUcastPkts: Number of subnet unicast packets that the higher layer protocols requested to be transmitted to a subnet unicast address including those that were discarded or not sent.

ifOutNUcastPkts: Number of subnet non-unicast packets that the higher layer protocols requested to be transmitted to a subnet non-unicast (ie. broadcast or multicast) address including those that were discarded or not. This variable is not supported by all agents. In agents where it is not supported, non-unicast packets are counted together with unicast packets.

ifOutDiscards: Number of outbound packets which were chosen to be discarded even though no errors had been detected. Eg. to free up buffer space when the output buffers are full.

ifOutErrors: Number of outbound packets that could not be transmitted because of errors.

A.2. PROTEON SPECIFIC SNMP VARIABLES

A.2.1. Proteon Generic Variables

The Proteon Generic Interface variables are mandatory for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

interface-type: The type of interface according to the link level protocol. These include:

ProNET-10/80 Token Ring
Interlan, Proteon, & Western Digital Ethernet
COM-2 (T1) & COM-4 (56 kbps) serial lines
ProNET-4 IEEE 802.5 Token Ring
HYPERchannel

maint-failed: Total number of maintenance check failures on an interface. Maintenance checks are carried out at regular intervals and a pre-specified number of maintenance test failures are allowed before a more comprehensive self-test is performed.

selftest-passed: Total number of self-test successes on this interface.

selftest-failed: Total number of self-test failures on an interface. Self-tests are carried out at regular intervals or when a pre-specified number of maintenance check failures occur.

A.2.1. Proteon Interface Variables

A.2.1.1. ProNET-10/80 Interface Variables

The Pronet-10/80 variables are optional for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

data-present: The number of times the unread data was present in the input packet buffer.

overrun: The number of times that a packet copied from the ring exceeded the size of the input packet buffer.

odd-byte-count: The number of times that a packet with an odd byte length was received.

parity-error: The number of times that a parity error was detected in a packet copied from the ring.

bad-format: The number of times that a formatting error was detected in a packet copied from the ring.

not-in-ring: The number of times that the relays in a proNET wire center were found to be unenergised.

out-ring-inits: The number of ring initializations attempted.

out-bad-format: The number of packets destined for the ring that were determined to be improperly formatted. This is incremented when an incorrectly formatted packet was encountered by the interface as it drained the message from the ring during an output operation. It is also incremented if the interface was unable to recognize its returning message on the ring.

out-timeout: The number of times that an attempt to put a packet onto the ring was delayed for more than 700 ms.

A.2.1.2. Interlan Ethernet Variables

The InterLAN Ethernet variables are optional for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

in-overruns: The number of times that a frame copied from the line was greater than the maximum permissible frame size.

in-dropped: The number of frames that were detected but lost.

out-retry: The number of times that an output operation was retried due to a transmission failure.

out-failed: The number of times that an output failure occurred.

out-excess-coll: The number of times that a frame destined for the line incurred 16 collisions on the line. This indicates extroadinarily high network load, or a hardware fault somewhere on the network.

frag-rcvd: The number of collision fragments and short packets received.

frames-lost: The number of frames lost due to insufficient buffer space.

mcast-accept: The number of frames received which were addressed one of the controller's multicast addresses.

mcast-reject: The number of frames received which were not addressed to one of the controller's multicast addresses.

crc-errors: The number of times that a frame received contained a CRC error. These are counted internally by a chipset, which only interrupts when one of the counters overflows. Thus, this value may not always be current, but will be within 16 of the true value.

align-errors: The number of frames received that were misaligned. A frame is misaligned if it's received length is not a multiple of 8 bits. These are counted internally by a chipset, which only interrupts when one of the counters overflows. Thus, this value may not always be current, but will be within 16 of the true value.

total-out-coll: The number of times that a frame in the process of being put on the line collided with another frame.

out-window-coll: The number of times that a frame in the process of being put on the line collided with another frame after the first 51.2 microseconds of transmission (one slot time). This may indicate that some other node on the network is failing to defer, or the network is too long.

A.2.1.3. Proteon/Western Digital Ethernet Variables

The Proteon/Western Digital Ethernet variables are optional for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

spurious-interrupts: The number of times the interrupt sense routine was entered but there was no ethernet events to process.

in-overruns: The number of received frames whose length was greater than the maximum ethernet frame size.

in-dropped: The number of times that the adapter asserted an override warning, which is posted when the received buffer is full. It does not necessarily mean that a frame is lost.

frames-lost: The number of frames lost due to insufficient buffer space.

crc-errors: The number of times that a frame received contained a CRC error. These are counted internally by a chipset, which only interrupts when one of the counters overflows. Thus, this value may not always be current, but will be within 16 of the true value.

align-errors: The number of frames received that were misaligned. A frame is misaligned if its received length is not a multiple of 8 bits. These are counted internally by a chipset, which only interrupts when one of the counters overflows. Thus, this value may not always be current, but will be within 16 of the true value.

in-fifo-overrun: The number of times the input FIFO in the adapted overflowed while copying a frame from the line.

in-multicasts: The number of frames received which were addressed to one of the controller's multicast addresses.

jabber-conditions: The number of times that the transceiver has continuously asserted the collision detect line, which its way of signalling that it has detected a jabber condition (unending transmission). This indicates a serious hardware failure, which may be jamming the ethernet.

out-success-w/retry: The number of frames which were successfully transmitted after one or more collisions.

out-excess-coll: The number of times that a frame destined for the line incurred 16 collisions and was discarded. This indicates extroadinarily high network load, or a hardware fault somewhere on the network.

out-carrier-sense-lost: The number of times that the transmit carrier sense was lost.

out-fifo-underrun: The number of times the transmission of a packet failed due to the inability to fetch bytes from the local FIFO packet buffer fast enough to transmit on the network.

out-cd-heartbeat-error: The number of times that a heartbeat was not detected after transmitting a frame. The packet will be treated as sucessfully transmitted, since some transceivers do not generate heartbeat.

out-window-coll: The number of times that a frame in the process of being put on the line collided with another frame after the first 51.2 microseconds of transmission (one slot time). This may indicate that some other node on the network is failing to defer, or the network is too long.

A.2.1.4. COM-2/COM-4 Serial Interface Variables

The COM-2 (T1) and COM-4 (56kbps) variables are optional for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

xmit-framing-errors: (COM-2 only) The number of framing errors which occurred during transmission. Indicates that the router attempted to send a packet less than 2 bytes long. This would be a software error.

xmit-underruns: The number of underrun errors which occurred during transmission. This indicates that the DMA controller on the interface did not provide data to the USART fast enough on a transmit. The transmitted frame is aborted.

xmit-no-dcd-errors: The number of transmissions which failed due to the absence of the Data Carrier Detect signal.

xmit-no-cts-errors: The number of transmissions which failed due to the absence of the Clear To Send signal.

xmit-no-dsr-errors: The number of transmissions which failed due to the absence of the Data Set Ready signal.

recv-framing-errors: The number of HDLC framming errors which occurred during reception. This could be because the packet was not a multiple of 8 bits long or a bit stuffing violation. Usually caused by line noise.

recv-overruns: The number of overrun errors which occurred during reception. This indicates that the DMA controller on the interface was not removing data from the USART fast enough. This also happens when a packet larger than 2048 bytes is received. Overlength packets can be caused by line errors.

recv-aborts: The number of aborted frames received. A HDLC abort was received in the middle of the frame. Either the other end of the line aborted a transmission, or there was a bit error on the line that looked like an abort.

recv-crc-errors: The number of received with HDLC CRC-16 errors. This indicates a line error.

recv-buffer-overflows: The number of received frames whose length was greater than the maximum allowed. This indicates that the interface is unable to move received packets from local memory to the gateway's memory fast enough. This would probably indicate that there is too much DMA activity on the Multibus.

recv-buffer-locked: The number of lost due to a lack of buffer space. This indicates that the interface is unable to move received packets from local memory to the gateway's memory fast enough. This would probably indicate that there is too much DMA activity on the Multibus. This counter indicates that the interface is further behind in copying packets than the "recv-buffer-overflow" error.

modem-signals: An indication of the state of certain modem signals.

A.2.1.5. ProNET-4 Token Ring Variables

The Pronet-4 variables are optional for all Proteon systems that support SNMP. These variables are maintained on a "per interface" basis.

current-ring-state: The current ring state with respect to the interface.

adapter-check-state: The adapter check state which has a value of zero until an adapter check occurs.

signal-losses: The number of times that the adapter detected a loss of signal on the ring.

hard-errors: The number of the adapter was transmitting or receiving beacon frames.

transmitting-beacons: The number of beacon frames transmitted by the adapter.

lobe-wire-faults: The number of times the adapter detected an open or short circuit in the lobe data path.

auto-removal-errors: The number of times the adapter has detected a hardware error following a Beacon Auto-removal.

remove-receiveds: The number of Remove Ring Station MAC frame requests received.

ring-recovery-actions: The number of times the adapter was transmitting or receiving Claim Token MAC frames.

line-errors: The number of times any of the following conditions are detected: (i) a frame was repeated or duplicated, (ii) the EDI was zero, (iii) a code violation existed between the SDEL and EDEL, (iv) a code violation existed within a token, or (v) an FCS error existed.

burst-errors: The number of times the adapter detected the absence of transitions for five half bit times between the SDEL and EDEL.

lost-frames: The number of times the adapter failed to receive the end of the frame it was transmitting.

receive-congestions: The number of times the adapter, in repeat mode, recognized a frame with its specific address but had no buffer space into which the frame could be copied.

frame-copy-errors: The number of times the adapter recognized a frame with its specific address but finds the ARI bits set to 00.

token-errors: The number of times the adapter detected any of the following conditions: (i) a token with non-zero priority and monitor count bit set to 1, (ii) a frame and the monitor count bit is set to 1, (iii) no token or frame received with 10ms, or (iv) the starting delimiter/token sequence had a code violation.

DMA-bus-errors: The number of times an error was detected on the internal DMA bus.

DMA-parity-errors: The number of parity errors detected on the internal DMA bus.

ARI/FCI-errors: The number of times the adapter detected that the upstream neighbor station was unable to set the ARI or FCI bits in received frames.

B. SNMPLOG

B.1. SNMPLOG UNIX MAN PAGE

NAME

snmplog - SNMP querying application for multiple agents

SYSNOPSIS

snmplog <configfile>

DESCRIPTION

snmplog is a generic querying application designed to execute on a host and query any number of monitoring agents for information over an extended period of time. What information is requested, from which agents, and how often the information is requested are all dictated by the snmplog configuration file (described below).

snmplog is an application process that allows for querying granularity at the level of both agents and variables by means of specifying multipliers to a user configurable base interval (see configuration file description below). So, for example, with a base interval of X seconds, querying of a given agent with agent interval multiplier Y, for a variable with variable interval multiplier Z, under a MIBxxx or PRVxxx statement with interval multiplier W, will occur every X * Y * Z * W seconds. Currently, snmplog output can be logged into a UNIX flat file.

SNMPLOG CONFIGURATION FILE

The snmplog configuration file is consulted upon startup of an snmplog process. The statements in the configuration file are used to determine what variables to query from what agents, how often the queries are performed, and where to put the responses to the queries. The general format of the configuration file is a set of global controls followed by a set of group definitions followed by a set of agent definitions.

The global controls consist of BASE, PIDFILE, and LOGFILE statements. A group definition consists of a GROUP statement followed by a set of polling definitions. A polling definition consists of MIBxxx and PRVxxx statements, each followed by LOG statements specifying what variables are to be queried in the circumstances specified by the MIBxxx and PRVxxx statements. The agent definitions consist of a list of AGENT statements.

The statements in the configuration file are:

BASE <base-interval>

Sets the base querying interval to <base-interval> seconds for this invocation of the snmplog process. The intervalis specified in seconds. This is the only statement that is mandantory.

PIDFILE <pid-filename>

Store this snmplog process's process ID (PID) in the UNIX flat file <pid-filename>. This file is overwritten if the next invocation of snmplog specifies the same file name. If no PIDFILE statement is present, the PID is not written into a file.

LOGFILE <log-filename>

The output from snmplog will be written to the UNIX flat file <log-filename>. All output from snmplog will be logged into this logfile if it is specified. If no LOGFILE statement is present, output will not be logged to a UNIX flat file.

GROUP <group>

Associate <group> with subsequent MIBxxx, PRVxxx, and LOG statements.

MIBSYS <multiplier>

Execute following LOG lines for instance 0 for every AGENT associated with the current <group>. That is, the following variables are logged on a per AGENT rather than per interface basis.

Note: If a group definition includes MIBIFx or PRVIFx statements, _mgmt_mib_numIntf must be queried by a LOG statement associated with a MIBSYS statement in the group definition.

Note: If a group definition includes PRVxxx statements, _mgmt_mib_sysObjId must be queried by a LOG statement associated with a MIBSYS statement in the group definition.

PRVSYS <multiplier> <sysObjId-prefix>

Execute following LOG lines for instance 0 for every AGENT associated with the current <group> whose sysObjId is prefixed by <sysObjId-prefix>.

MIBIF <multiplier> [<type-ObjId>]

Execute following LOG lines for every interface of every AGENT associated with the current <group>. If specified, the value of variable <type-ObjId> will be retained for subsequent use by MIBIFT statements.

Note: If a group definition includes MIBIFT statements, <type-ObjId> must be specified and <type-ObjId> must be queried by a LOG statement associated with a MIBIF statement in the group definition.

MIBIFT <multiplier> <value>

Execute following LOG lines for every interface whose <type-ObjId> value matches <value> of every AGENT associated with the current <group>.

PRVIF <multiplier> <sysObjId-prefix> [<type-ObjId>]

Execute following LOG lines for every interface of every AGENT associated with the current <group> whose sysObjId is prefixed by <sysObjId-prefix>. If specified, the value of variable <type-ObjId> will be retained for subsequent use by PRVIFT statements.

Note: If a group definition includes PRVIFT statements, <type-ObjId> must be specified and <type-ObjId> must be queried by a LOG statement associated with a PRVIF statement in the group definition.

PRVIFT <multiplier> <sysObjId-prefix> <value>

Execute following LOG lines for every interface whose <type-ObjId> value matches <value> of every AGENT associated with the current <group> whose sysObjId is prefixed by <sysObjId-prefix>.

. LOG <variable> <log-multiplier>

Query the variable <variable> every <log-multiplier> * <multiplier> * <agent-multiplier> * <base-interval> seconds.

Names of variables to be logged should be specified beginning after '_iso_org_dod_internet': that is, generic MIB variables should be specified beginning with _mgmt_mib whereas enterprise specific variables should be specified beginning with _private.

AGENT <addr> <agent> <group> <comm> <agent-multiplier> <tmo> <try> <vpp>

Query the monitoring agent at IP address <addr> using the community name <comm>. Queries should be based on group definition <group>.

During queries, set the query timeout interval (after which retransmission occurs) to <tmo> seconds. Query retransmissions will be retried <try> times. The number of variables that the monitoring agent can pack into a reply packet is <vpp>.

LOG FILE FORMAT

Every time a variable is queried, its value is recorded in <log-filename>. The format of the log file is:

```
<agent> <instance> <time-stamp> <var-objld> <var-value>
```

where:

<agent>	is from the AGENT statement
<instance>	is the object instance (eg. the interface number)
<time-stamp>	is the UNIX timestamp of the query
<var-oldld>	is a representation of the SNMP object ID
<var-value>	is a appropriate representation of the object value

Examples:

```
acerl     0     639895823    _2.1.1.2            _4.1.1.1.1.42
```

The value of AGENT acerl's _mgmt_mib_system_sysObjectID at Wed Apr 11 23:50:23 CDT 1990 was _4.1.1.1.1.42, which is equivalent to _private_enterprise_proteon_admin_1.42.

```
acerl     1     639895824    _2.1.2.2.1.16      767325999
```

On Wed Apr 11 23:50:24 CDT 1990, the value of AGENT acerl's _mgmt_mib_interfaces_ifTable_ifEntry_ifOutOctets for its first interface was 767325999.

FILES

/etc/snmp.variables	The variable initfile
<configfile>	The snmplog configuration file
<pid-filename>	File containing PID of currently running snmplog process
<log-filename>	File to log variable values

SEE ALSO

"Simple Network Management Protocol", RFC-1098, by J. D. Case, M. S. Fedor, M. L. Schoffstall and J. R Davin; "SNMP Implementation 3.3", by H. M. Tam, M. L. Schoffstall, W. Yeong, M. S. Fedor; "Management Information Base for Network

Management of TCP/IP Based Internets", RFC 1066, by M. Rose and K. McCloghrie; and "Structure and Identification of Management Information for TCP/IP Based Internets", RFC-1065, by M. Rose and K. McCloghrie.

B.2. SAMPLE SNMPLOG CONFIGURATION FILE

```
#
# snmplog.cf
# snmplog configuration file.
#
#  Syntax:
#   BASE intvl (in seconds)
#   LOGFILE logfilename
#   PIDFILE pidfilename
#   GROUP groupname
#    MIBSYS imult
#     PRVSYS imult sysObjID-prefix
#     MIBIF imult [type-objID]
#     PRVIF imult sysObjID-prefix [type-objID]
#     MIBIFT imult type-objID-value
#     PRVIFT imult sysObjID-prefix type-objID-value
#    AGENT a.b.c.d sitename groupname communityid imult timeout(in sec) retries numvar
#    LOG variablename imult
#

BASE 900

# Log file path configured for seka
PIDFILE  /usr/adm/snmp/snmplog.pid
LOGFILE  /usr/adm/snmp/data/active.log

#
# Agents other than CICnet
#

GROUP normal

MIBSYS 1
LOG _mgmt_mib_system_sysObjectID 4
LOG _mgmt_mib_system_sysUpTime 4
LOG _mgmt_mib_interfaces_ifNumber 4

        <some LOG lines deleted here>

MIBIF 1
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifType  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifAdminStatus  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOperStatus  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInErrors  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutErrors  4

#
# EN641s

PRVIF  1  _3.1.10
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInOctets  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInNUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInDiscards  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutOctets  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutNUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutDiscards  4

#
# Proteon agents

PRVIF  1  _private_enterprises_proteon
_private_enterprises_proteon_xface_generic_interface-type
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInOctets  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInNUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifInDiscards  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutOctets  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutNUcastPkts  4
LOG  _mgmt_mib_interfaces_ifTable_ifEntry_ifOutDiscards  4
LOG  _private_enterprises_proteon_xface_generic_interface-type  4
LOG  _private_enterprises_proteon_xface_generic_maint-failed  4
LOG  _private_enterprises_proteon_xface_generic_selftest-passed  4
LOG  _private_enterprises_proteon_xface_generic_selftest-failed  4

# ProNET-10
PRVIFT  1  _private_enterprises_proteon  2
LOG  _private_enterprises_proteon_xface_pronet_data-present  4
LOG  _private_enterprises_proteon_xface_pronet_overrun  4
LOG  _private_enterprises_proteon_xface_pronet_odd-byte-count  4
LOG  _private_enterprises_proteon_xface_pronet_parity-error  4
LOG  _private_enterprises_proteon_xface_pronet_bad-format  4
LOG  _private_enterprises_proteon_xface_pronet_not-in-ring  4
LOG  _private_enterprises_proteon_xface_pronet_out-ring-inits  4
LOG  _private_enterprises_proteon_xface_pronet_out-bad-format  4
LOG  _private_enterprises_proteon_xface_pronet_out-timeout  4

          <some LOG lines deleted here>

# Interlan Ethernet
PRVIFT  1  _private_enterprises_proteon  8
LOG  _private_enterprises_proteon_xface_ieth_in-overruns  4
LOG  _private_enterprises_proteon_xface_ieth_in-dropped  4
LOG  _private_enterprises_proteon_xface_ieth_out-retry  4
LOG  _private_enterprises_proteon_xface_ieth_out-failed  4
LOG  _private_enterprises_proteon_xface_ieth_out-excess-coll  4
LOG  _private_enterprises_proteon_xface_ieth_frag-rcvd  4
LOG  _private_enterprises_proteon_xface_ieth_frames-lost  4
LOG  _private_enterprises_proteon_xface_ieth_mcast-accept  4
LOG  _private_enterprises_proteon_xface_ieth_mcast-reject  4
LOG  _private_enterprises_proteon_xface_ieth_crc-errors  4
LOG  _private_enterprises_proteon_xface_ieth_align-errors  4
LOG  _private_enterprises_proteon_xface_ieth_total-out-coll  4
LOG  _private_enterprises_proteon_xface_ieth_out-window-coll  4

          <some LOG lines deleted here>

# COM-2 Serial Line
PRVIFT 1 _private_enterprises_proteon 10
LOG  _private_enterprises_proteon_xface_comsl_xmit-framing-errors  4
LOG  _private_enterprises_proteon_xface_comsl_xmit-underruns  4
LOG  _private_enterprises_proteon_xface_comsl_xmit-no-dcd-errors  4
LOG  _private_enterprises_proteon_xface_comsl_xmit-no-cts-errors  4
LOG  _private_enterprises_proteon_xface_comsl_xmit-no-dsr-errors  4
LOG  _private_enterprises_proteon_xface_comsl_recv-framing-errors  4
LOG  _private_enterprises_proteon_xface_comsl_recv-overruns  4
LOG  _private_enterprises_proteon_xface_comsl_recv-aborts  4
LOG  _private_enterprises_proteon_xface_comsl_recv-crc-errors  4
LOG  _private_enterprises_proteon_xface_comsl_recv-buffer-overflows  4
LOG  _private_enterprises_proteon_xface_comsl_recv-buffer-locked  4
LOG  _private_enterprises_proteon_xface_comsl_modem-signals  4

          <some LOG lines deleted here>

#
# Cisco agents

PRVIF 1 _private_enterprises_cisco
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifInOctets 4
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifInNUcastPkts 4
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifInDiscards 4
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifOutOctets 4
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifOutNUcastPkts 4
LOG _mgmt_mib_interfaces_ifTable_ifEntry_ifOutDiscards 4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Runts  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Giants  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_CRC  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Frame  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Overrun  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Ignored  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Abort  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Resets  4
LOG  _private_enterprises_cisco_local_if_Table_Entry_Restarts  4

                          <some GROUP lines deleted here>

#
# Agents to be polled
#

AGENT 128.174.55.31 admin normal public 1 2 2 8
AGENT 128.174.7.6 allen normal public 1 2 2 8
AGENT 128.174.7.46 altgeld normal public 1 2 2 8
AGENT 128.174.7.47 asl normal public 1 2 2 8
```

BEST COPY AVAILABLE

```
AGENT 128.174.3.8 atmsci normal public 1 2 2 8
AGENT 128.174.55.134 bmri normal public 1 2 2 8
AGENT 128.174.100.110 ce normal public 1 2 2 8
AGENT 128.174.7.51 commw normal public 1 2 2 8
AGENT 128.174.7.35 commw2 normal public 1 2 2 8
AGENT 128.174.100.19 csl normal public 1 2 2 8
AGENT 128.174.100.134 daven normal public 1 2 2 8
AGENT 128.174.7.3 daven2 normal public 1 2 2 8
AGENT 128.174.3.30 dcl normal public 1 2 2 8
AGENT 128.174.5.113 dcl3 normal public 1 2 2 8
AGENT 128.174.100.240 dcl4 normal public 1 2 2 8
AGENT 128.174.55.130 dcl5 normal public 1 2 2 8
AGENT 128.174.3.10 dcl6 normal public 1 2 2 8
AGENT 128.174.55.2 dcl7 normal public 1 2 2 8
AGENT 128.174.7.49 dkh normal public 1 2 2 8

                    <some AGENT lines deleted here>
```

B.3. SAMPLE DAILY REPORTS

B.3.1. Fault and Traffic Report

A sample Daily Fault and Traffic report for the monitored agents is given below. The various categories reported are:

If#	Interface number to which the parameters on this line belong to.
ID	Interface identifier, usually the subnet number of the interface.
Type	Type of interface.
Fail	Number of Proteon maintenance test (MT) and self-test (ST) failures for this interface.
Error	Number of Generic SNMP Input and Output Errors for this interface.
Discard	Number of Generic SNMP Input and Output Discards for this interface.
UcastKpkt	Number of Generic SNMP Input and Output Unicast packets (in thousands) for this interface.
NUcastKpkt	Number of Generic SNMP Input and Output Non-Unicast packets (in thousands) for this interface (not supported by Proteon currently).
MBytes	Number of Generic SNMP Input and Output Megabytes for this interface.
Sample Interval	Time interval over which the displayed data was measured (in the form <hours>: <minutes>)
@	Variables on this line fall above the threshold denoted by the corresponding symbol: S: (self) the interface's own threshold. G: (group) the threshold of the statistical group to which the interface belongs. B: (both) both the above thresholds.

```
                         SNMP SUMMARY REPORT

Data per Gateway Interface: Mon Apr  2 23:50:12 1990 - Tue Apr  3 23:52:36 1990

Agent:  admin       128.174.55.31      PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 1   7    Pr10   0  0     46  80   0    0   1067  1814     0     0   222   219   24:00
   @ G       46  ifInErrors                  S       80  ifOutErrors
   @ G       43  bad-format                  S       81  out-bad-format

                  <some agents deleted here>

Agent:  daven       128.174.100.134    PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 1   7    Pr10   0  0     32  11   0    0   2711  2795     0     0   230   230   24:00
   @ B       32  ifInErrors                  B       31  bad-format
 2  183   IEth   0  0      0   0   0    0     80    71     0     0     6     8   24:00
   @ S        2  total-out-coll              S        2  out-window-coll
   @ S        2  out-retry                   S        1  frag-rcvd

Agent:  dcl         128.174.3.30       PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 1   3    Pr10   0  0      3   0   0    0    979   514     0     0   159    85   24:00
   @ S        3  ifInErrors                  B        2  overrun

Agent:  dcl2        130.126.174.8      PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 2   -    Com4   0  0      0   0   0    0    797   679     0     0   120    83   24:00
   @ G        3  recv-crc-errors

Agent:  dcl3        128.174.5.113      PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 1   3    Pr10   0  0      0   5   0    0   1010  1176     0     0   126   166   24:00
      @      S      1  odd-byte-count
 4   5    PEth   0  0      0   0   0    0   5061  4988     0     0  1068   464   24:00 +
   @ S       25  crc-errors                  S        9  align-errors

                  <some agents deleted here>

Agent:  dcl5        128.174.55.130     PROTEON P4200
If#  ID   Type   Fail     Error   Discard    UcastKpkt    NUcastKpkt   MBytes    Sample
                 MT ST    In Out  In  Out     In    Out    In   Out    In   Out  Interval
 4  33    PEth   0  0      1   0   0    0    570   787     0     0    64   329   24:00
   @ S        1  ifInErrors                  S        1  in-overruns

                  <some agents deleted here>

Agent:  dcl7        128.174.55.2       PROTEON P4200        1 Restarts
```

```
Agent:  eeb          128.174.100.111     PROTEON P4200
If#  ID    Type   Fail       Error   Discard     UcastKpkt    NUcastKpkt   MBytes      Sample
                 MT  ST      In  Out  In  Out       In    Out   In   Out      In   Out  Interval
3    114   PEth   0   0       0    0 21047 0    652    172     0     0      60    25   24:00
     @  G        21047   ifInDiscards           G         36486+  in-dropped
     @  G        63865+  frames-lost

                    <some agents deleted here>

Agent:  mossberg     128.174.5.8         CISCO TS
If#  ID    Type   Fail       Error   Discard     UcastKpkt    NUcastKpkt   MBytes      Sample
                 MT  ST      In  Out  In  Out       In    Out   In   Out      In   Out  Interval
1    5     Eth    0   0  117     0 43     0  1580   1668    58     0     153    98   24:00
     @  G        43  ifInDiscards               B         117   ifInErrors
     @  B        62  Runts                      B         29  CRC
     @  B        84  Frames                     B         117   ifInErrors

                    <some agents deleted here>

Agent:  wood         128.174.100.118     PROTEON P4200         1  Restarts
If#  ID    Type   Fail       Error   Discard     UcastKpkt    NUcastKpkt   MBytes      Sample
                 MT  ST      In  Out  In  Out       In    Out   In   Out      In   Out  Interval
1    43    lEth   0   0       0    0   0    0    909    868     0     0     142   132   24:00 +
     @  G        7230+   crc-errors             S         220+    total-out-coll
     @  S        46+   out-window-coll          S         219+    out-retry

+ Figures for this interface cover a period less than the indicated sample interval.

NB:  Counters for Non-Unicast Packets are supported in only certain types of gateways.

                    <some domains deleted here>

SUMMARY DATA FOR DOMAIN campus

Type      #If   Fail     Error     Discard      UcastKpkt    NUcastKpkt      MBytes
                MT  ST   In  Out    In   Out      In    Out    In   Out      In    Out
Eth       140    5   6   29   42 21185    0  47877  42991   872 1751   11026  10353
802.5     8      0   0    0    0     0    0    224     92     0    0      22     17
HChn      8      0   0    0    0     0    0  22102  22514     0    0     124   8918
          Total  5   6   29   42 21185    0  70203  65598   872 1751   11172  19289

Total number of agents included in these totals:  78
Total number of interfaces included in these totals:  156

SUMMARY DATA FOR DOMAIN backbone

Type      #If   Fail     Error     Discard      UcastKpkt    NUcastKpkt      MBytes
                MT  ST   In  Out    In   Out      In    Out    In   Out      In    Out
Eth       4      0   0    0    0     0    0   4123   4044   119   39     837    791
Pr10      32     0   0  735  303  2116    0  17640  11606     0    0    2666   1290
Pr80      40     0   0    0  503    14  138  38513  20857     0    0    8402   2966
Ser       12    10  10    0    0     0    0    878    619     0    0     116    115
          Total 10  10  735  806  2130  138  61155  37128   119   39   12023   5163
Total number of agents included in these totals:  73
Total number of interfaces included in these totals:  88

SUMMARY DATA FOR VARIOUS SUBNET

Subnet    #If    Fail     Error     Discard      UcastKpkt   NUcastKpkt     MBytes
               MT  ST    In  Out    In   Out     In     Out    In   Out     In    Out
.3         6    0   0     4   24     0     0   3778    3023     0     0    621    474
.7        26    0   0   731  279  2116     0  13861    8583     0     0   2045    866
.55       13    0   0     0   57    14     0   9815    6267     0     0   2176   1140
.100      27    0   0     0  446     0   138  28697   14590     0     0   6226   1826
.10        4    0   0     0    0     0     0  12284   12696     0     0     97   4703
.23        3    0   0     0    0     0     0    827     680   143    17    153    117
.121       2    0   0     0    0     0     0    379    4434    22     5     73    176
.211       4    0   0     0    0     0     0   9817    9818     0     0     26   4215
.235       4    0   0     0    0     0     0    809     826    88     8    121     96
.252       5    0   0     0    0     0     0   9944    6777   142     8   1906   1317
.5         5    0   0   130    3    72     0   9608    9183   155   798   1777   1023
.89        4    0   0     0    0     0     0   4123    4044   119    39    837    791
.174       6    0   010769    3     1     0  24830   24833    28     5   2519   2644

Total number of interfaces included in these totals:  109
```

B.3.2. CICnet Link Usage Report

A sample Daily Link Usage report for the CICnet links is given below.

```
                         DAILY LINK USAGE

Reporting Interval: Mon Apr  2 23:51:49 1990 - Tue Apr  3 23:52:32 1990

Link            Line      Agent:  Average     --------   Peak    --------
                Speed     InterfaceUsage        Usage    Ending   Interval
                (bps)     In/Out    (%)          (%)     Time      (sec)

iu -> osu       1.536M    iu:2      0.51         1.32    10:20:13   900
                          osu:2     0.50         1.28    10:20:14   900
osu -> iu       1.536M    osu:2     1.41        20.50    10:05:14   765
                          iu:2      1.37        20.17    10:05:13   771

uiuc -> iu      1.536M    uiuc:5    1.22         5.11    20:52:29   1033
                          iu:3      1.17         5.00    20:52:13   1020
iu -> uiuc      1.536M    iu:3      1.81        20.42    10:05:13   771
                          uiuc:5    1.77        20.34    10:05:16   756

                     <some links deleted here>

uia -> uiuc     1.536M    uia:3     2.29        12.47    14:35:15   900
                          uiuc:2    2.26        12.44    14:35:16   900
uiuc -> uia     1.536M    uiuc:2    5.28        28.74    20:20:15   899
                          uia:3     5.05        28.12    20:20:14   899
```

uoc -> uic	1.536M	uoc:4	2.67	11.87	17:35:19	900
		uic:2	2.65	11.76	17:35:15	899
uic -> uoc	1.536M	uic:2	4.79	32.82	17:05:15	798
		uoc:4	4.67	32.92	17:05:19	783
uoc -> uiuc	1.536M	uoc:5	3.79	9.83	11:35:19	900
		uiuc:3	3.73	9.87	11:35:16	899
uiuc -> uoc	1.536M	uiuc:3	3.09	12.73	17:35:16	899
		uoc:5	2.92	12.36	17:35:19	900
wisc -> umn	1.536M	wisc:4	1.73	11.84	14:35:20	900
		umn:2	1.61	11.53	14:35:17	899
umn -> wisc	1.536M	umn:2	3.13	26.51	16:20:18	899
		wisc:4	3.08	26.52	16:20:19	898

B.4. SAMPLE MONTHLY REPORTS

B.4.1. Traffic Report by Day

Sample graphs of the campus traffic (by day) for the month of March 1990 are shown in Fig B.1 and Fig B.2. The traffic here represents all the traffic into and out of the local departmental networks. The patterns generated reflect a typical week of five working days and a two day weekend (with Spring Break being from Mar 19 to Mar 25).

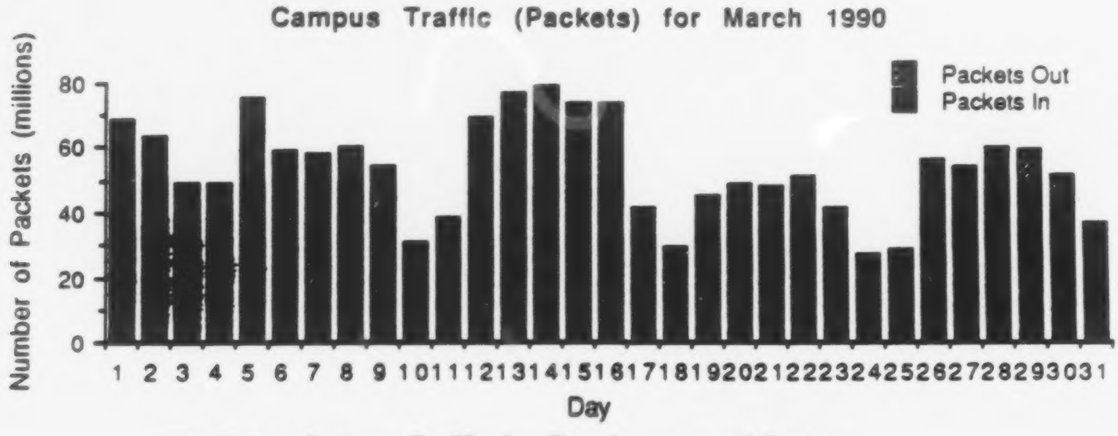


Fig B.1: Campus Traffic by Day in terms of Packets

It should also be noted that not all agents report "Bytes In" [Fig 3.2] correctly due to some bugs in their software. These agents tend to report much smaller values than expected.

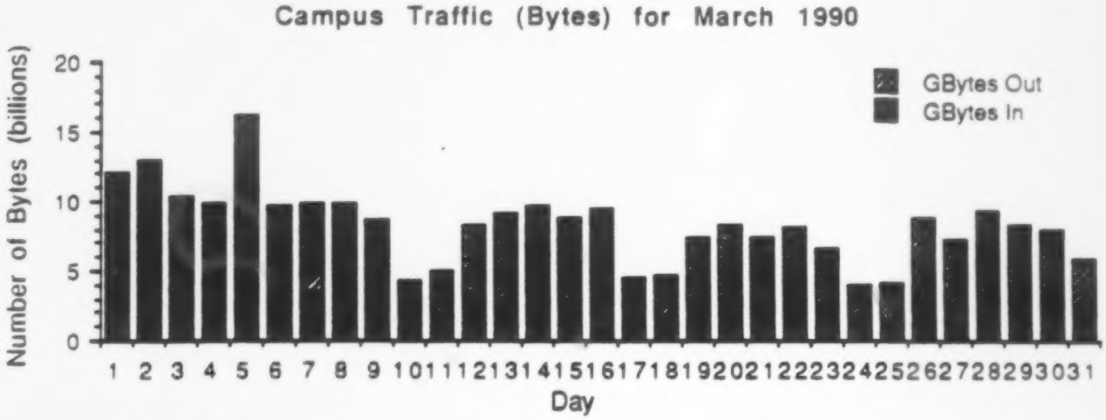


Fig B.2: Campus Traffic by Day in terms of Packets

B.4.2. Average Traffic Report by Month

Sample graphs of the Average Monthly Campus Traffic for the period November 1989 to March 1990 are shown in Fig B.3 and Fig B.4. It can be seen that the campus traffic reached a peak in January 1990, and then began to fall after that. This is because some of the departmental networks (eg. the Department of Computer Science) have been restructured so that their internal departmental traffic is kept off the campus backbone infrastructure, resulting in a decrease in the amount of traffic.

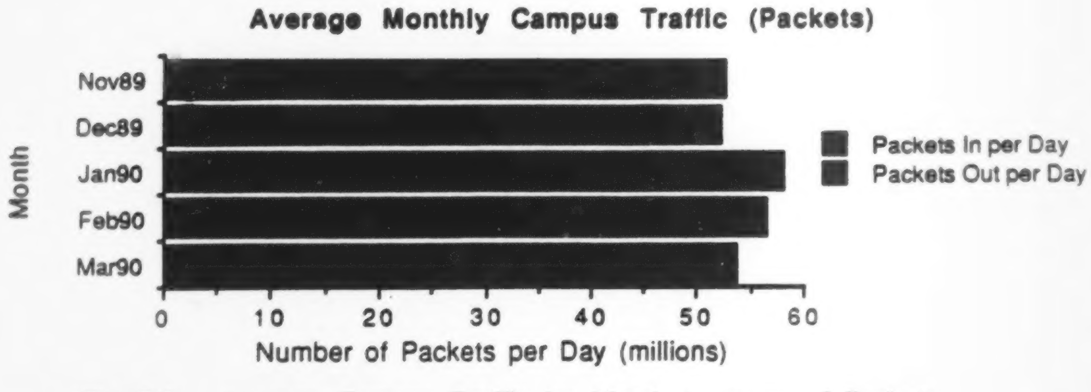


Fig B.3: Average Campus Traffic by Month in terms of Packets

Again, it should be noted that "GBytes In" tends to be under-reported, accounting for the slight unbalance between the averages shown in Fig 3.4.

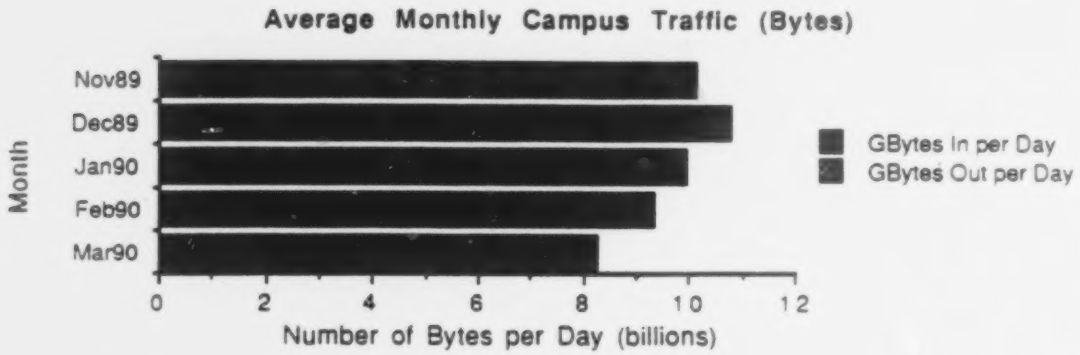


Fig B.4: Average Campus Traffic by Month in terms of Bytes

B.4.3. Link Usage Report

Sample graphs of the link usage for the link between the University of Iowa (UIA) and UIUC, are shown in Fig B.5 and Fig B.6. It can be seen that most of the time (36% of the time for traffic to UIUC and 23% for traffic to UIA), the link usage was at about 1% [Fig B.5]. Peak usage (48% and 24% usage for traffic to UIUC and UIA respectively) was achieved for a very small amount of time (0.03% in both cases).

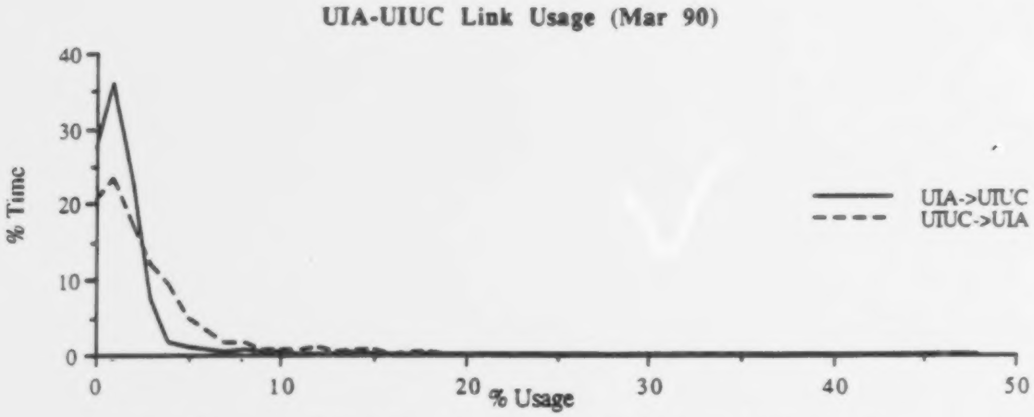


Fig B.6: Monthly Link Usage and Cumulative Link Usage

Fig B.6 shows that the link usage was below 5% for 95% of the time for traffic towards UIUC and 85% of the time for traffic towards UIA. So, although this T1 line is one of the busiest links in the CICnet backbone, it still has adequate bandwidth to meet its traffic demands.

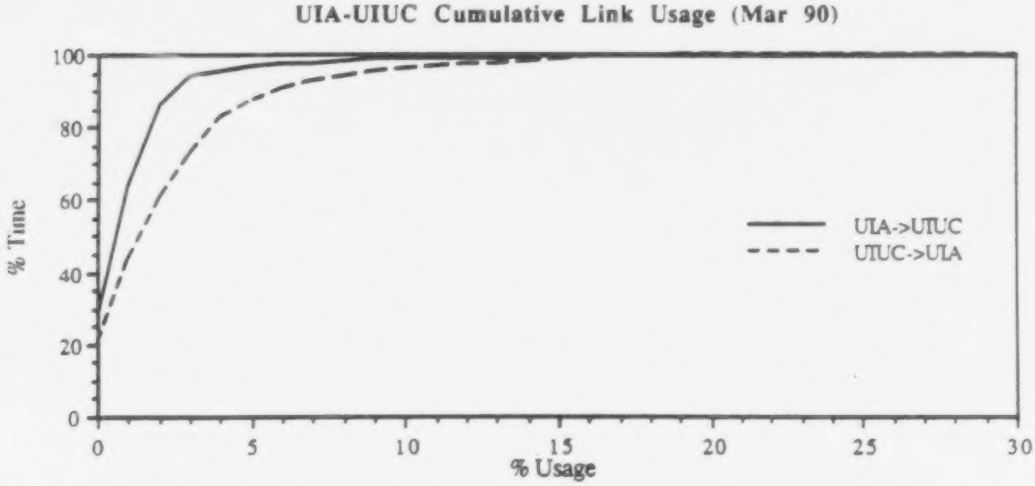


Fig B.6: Monthly Link Usage and Cumulative Link Usage

C. FAULT ISOLATION EXAMPLES

C.1. DECNET PROBLEMS ON ps.gw

C.1.1. Chronological Log of Events

Listed below is a chronological log of events concerning the 128.174.7 backbone ProNET-10 ring in Node 2 from late October 1989 to early December 1989. All the routers mentioned below are Proteon P4100 routers running Version 8.1a software.

Date	Event
Oct 23	ps.gw moved from a serial connection to the backbone via dcl7.gw to a direct connection to the 128.174.7 ring in Node 2.
Oct 24	DECnet Broadcast Routing Timer (BRT) on ps.gw changed from the default of 180 sec to 600 sec (to bring it in line with other P4200s running DECnet).
Nov 06	edu.gw installed on 128.174.7 ring in Node 2.
Nov 21	Fiber cable found to be tightly looped under ps.gw chassis.
Nov 27	Wrong type of loopback plug found on 128.174.7 ring wirecenter in Node 2.
Nov 29	DECnet disabled on ps.gw. It was found that when the BRT was set to a positive number other than default, the router returns a negative number when queried. Improperly tuned modem found on 128.174.7 ring in Node 2.
Dec 01	ps.gw temporarily replaced by a P4200 router running Version 8.1 software. Its BRT was set to 600 sec. Error counts dropped significantly.
Dec 14	ps.gw converted back to a P4100 router. Its BRT was set to default.

C.1.2. Wire Center Physical Connections

The physical connections of the 128.174.7.0 Ring wire center in Node 2, as of November 22, 1989, are shown in Fig C.1.

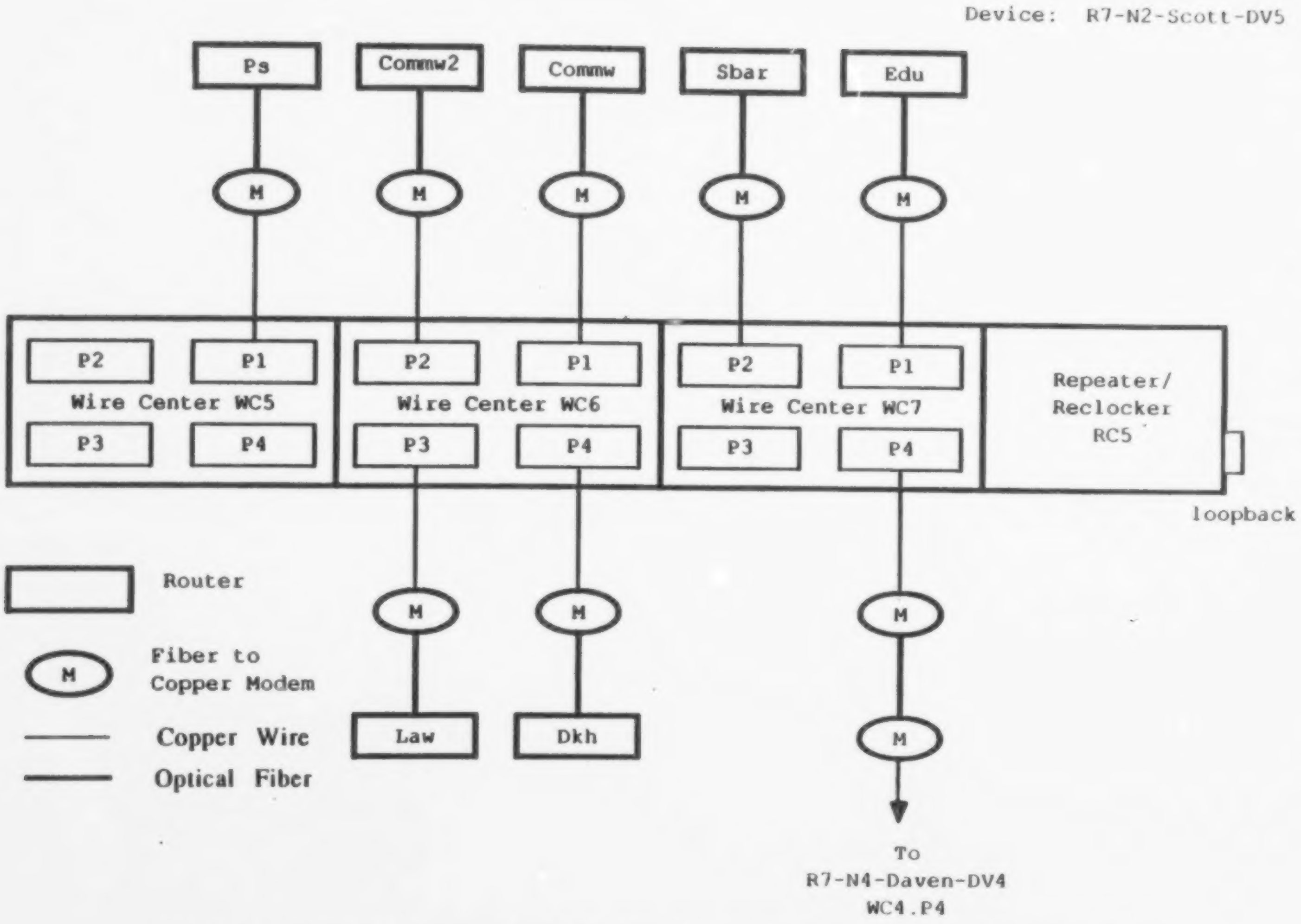
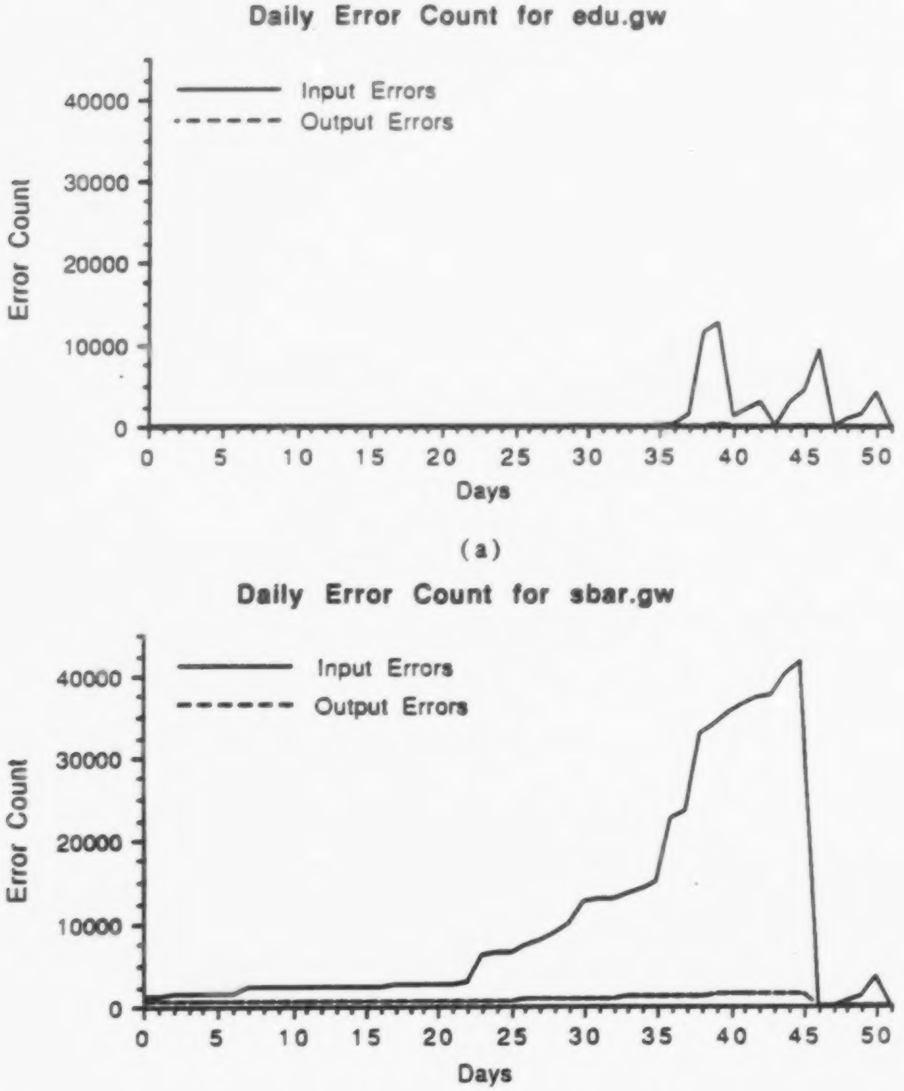

Device: R7-N2-Scott-DV5
Ps
Commw2
Commw
Sbar
Edu
M
M
M
M
M
P2
P1
Wire Center WC5
P3
P4
P2
P1
Wire Center WC6
P3
P4
P2
P1
Wire Center WC7
P3
P4
Repeater/
Reclocker
RC5
loopback
Router
M
Fiber to
Copper Modem
Copper Wire
Optical Fiber
M
M
M
M
Law
Dkh
To
R7-N4-Daven-DV4
WC4.P4

Fig C.1: Physical Connections of the 128.174.7 Ring Wire Center in Node 2

C.1.3. Input/Output Errors for P4100 Routers in Node 2

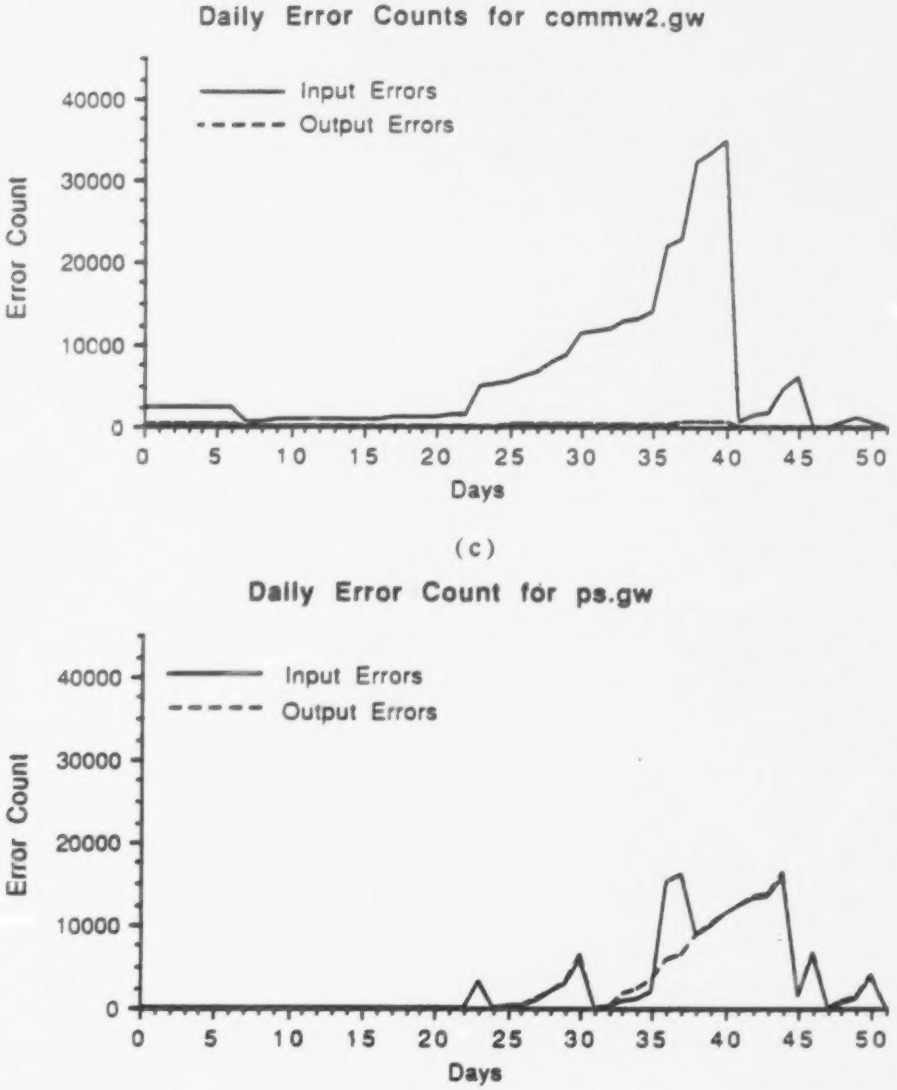
Fig C.2 shows the input and output errors for the P4100 routers attached to the 128.174.7.0 wire center in Node 2 during the period October 1, 1989, to November 22, 1989. Note that ps.gw was connected to the wire center on October 23, 1989, and edu.gw was connected to the wire center on Novermber 6, 1989.



(a)

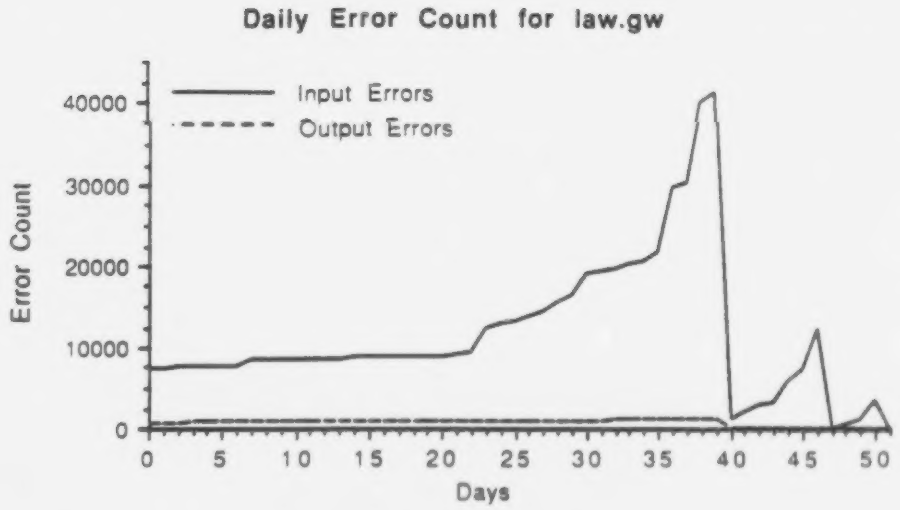
(b)

Fig C.2 (cont): Input and Output Errors (01-Oct-89 to 22-Oct-89)

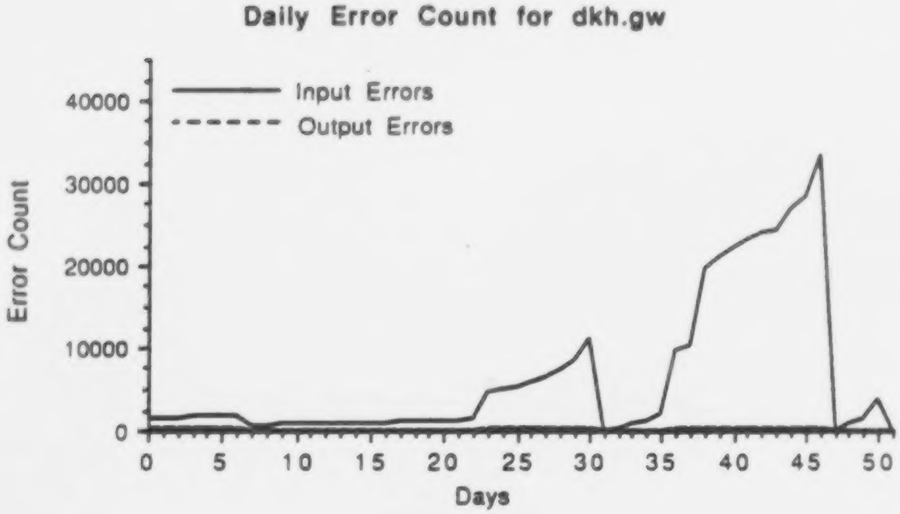


(d)

Fig C.2 (cont): Input and Output Errors (01-Oct-89 to 22-Oct-89)



(e)



(f)

Fig C.2 (cont): Input and Output Errors (01-Oct-89 to 22-Oct-89)

C.2. DOMAIN LOOP AT INDIANA UNIVERSITY

C.2.1. Extracts from the Daily Line Usage Report (Feb 22 & Feb 25)

```
                              DAILY LINE USAGE

Reporting Interval:  Wed Feb 21 23:54:23 1990 - Thu Feb 22 23:55:07 1990

Link          Line      Agent:       Ave        --------  Peak    --------
              Speed     Interface    Usage         Usage  Ending  Interval
              (bps)     In/Out       (%)          (%)     Time     (sec)

iu -> osu     1.536M     iu:2        0.75          2.31  13:54:29   3557
                        osu:2        0.74          2.30  13:54:36   3556
osu -> iu     1.536M    osu:2        1.90         16.41  11:55:22   3652
                         iu:2        1.85         16.13  11:55:15   3651

uiuc -> iu    1.536M    uiuc:5       2.52          5.85  22:54:54   3567
                         iu:3        2.30          5.14  22:54:36   3566
iu -> uiuc    1.536M     iu:3        2.45         16.67  11:55:15   3652
                        uiuc:5       2.34         16.52  11:55:34   3653
```

(a)

```
                              DAILY LINE USAGE

Reporting Interval:  Sat Feb 24 23:54:37 1990 - Sun Feb 25 23:55:04 1990

Link          Line      Agent:       Ave        --------  Peak    --------
              Speed     Interface    Usage         Usage  Ending  Interval
              (bps)     In/Out       (%)           (%)    Time     (sec)

iu -> osu     1.536M     iu:2         0.40         0.56  19:54:43   3601
                        osu:2         0.39         0.55  19:54:49   3601
osu -> iu     1.536M    osu:2         0.49         1.09  14:54:34   3580
                         iu:2         0.47         1.05  14:54:28   3581

uiuc -> iu    1.536M    uiuc:5        3.50         4.35  14:54:46   3579
                         iu:3         3.10         3.87  14:54:28   3580
iu -> uiuc    1.536M     iu:3        10.08        12.43  14:54:28   3580
                        uiuc:5        9.66        12.05  14:54:46   3579
```

(b)

Fig C.3: Extract from Daily CICnet Line Usage (Feb 22 & Feb 25)

C.2.1. Extracts from the Daily Fault and Traffic Report (Feb 22 & Feb 25)

```
Data per Agent Interface:  Wed Feb 21 23:50:11 1990 - Thu Feb 22 23:55:10 1990

Agent:  iu.cic.net     129.79.2.131        CISCO GS
If#  ID    Type    Fail       Error    Discard      UcastKpkt      NUcastKpkt    MBytes       Sample
                  MT  ST      In  Out   In  Out       In    Out    In    Out     In    Out   Interval
 1   IU     Eth    0   0     102    0  84    0   1013  3778   605     26    248    311     24:00
 2   OSU    Ser    0   0       2    0   0    0   1032   777    29      0    295    118     24:00
 3   UIUC   Ser    0   0       2    0   2    0   4109  1662    33      0    363    391     24:00

Agent:  uiuc.cic.net 130.126.174.10        CISCO GS
If#  ID    Type    Fail       Error    Discard      UcastKpkt      NUcastKpkt    MBytes       Sample
                  MT  ST      In  Out   In  Out       In    Out    In    Out     In    Out   Interval
 1   174    Eth    0   0    3413    0   1    0   9107  6057    12      3   1364   1048     24:00
 2   UIA    Ser    0   0     349    0   0    0   2857  2831    30      0    329    632     24:00
 3   UCC    Ser    0   0       6    0   0    0   2768  2776    32      0    548    508     24:00 +
 4   89     Eth    0   0       1    0   0    0      4   726    81     23     35     81     24:00
 5   IU     Ser    0   0      17    0   0    0   1633  4144    30      0    372    398     24:00 +
```

(a)

```
Data per Agent Interface:  Sat Feb 24 23:50:15 1990 - Sun Feb 25 23:55:06 1990

Agent:  iu.cic.net     129.79.2.131        CISCO GS
If#  ID    Type    Fail       Error    Discard      UcastKpkt      NUcastKpkt    MBytes       Sample
                  MT  ST      In  Out   In  Out       In    Out    In    Out     In    Out   Interval
 1   IU     Eth    0   0      62    0  35    0   5233  6636   555     29   1643     63     24:00
     +
 2   OSU    Ser    0   0       0    0   0    0    558   478    32      0     74    118     24:00
 3   UIUC   Ser    0   0       2    0   0    0   6785  5531    35      0    491   1595     24:00

Agent:  uiuc.cic.net 130.126.174.10        CISCO GS
If#  ID    Type    Fail       Error    Discard      UcastKpkt      NUcastKpkt    MBytes       Sample
                  MT  ST      In  Out   In  Out       In    Out    In    Out     In    Out   Interval
 1   174    Eth    0   0    1767    0   0    0  10027  6224    12      3   1113   1570     24:00 +
 2   UIA    Ser    0   0     163    0   1    0   1523  1383    32      0    159    249     24:00
 3   UCC    Ser    0   0       9    0   0    0   1882  1918    33      0    215    370     24:00
 4   89     Eth    0   0       0    0   0    0     40  2725    84     26     32    336     24:00
 5   IU     Ser    0   0       0    0   0    0   5499  6820    32      0   1528    553     24:00 +
```

(b)

+ Figures for this interface cover a period less than the indicated time used.

Fig C.4: Extract from Daily Fault and Traffic Report (Feb 22 & Feb 25)

GLOSSARY OF TERMS AND ABBREVIATIONS

address mask: A 32-bit mask used to select bits from an IP address for subnet addressing [Com88, MoP85, Mog84a].

ARP: (Address Resolution Protocol) This protocol allows a host to find the physical address of a target host on the same physical network, given the target host's Internet address. The source host broadcasts a special packet that asks the host with the target Internet address to respond with its physical address. The source host can then store the target host's physical address in an Address Resolution Cache for future use [Com88].

ARPAnet: The pioneering national network funded by ARPA (Advanced Research Projects Agency) which served as the basis for early networking research as well as a central backbone during the development of the Internet. The ARPAnet consists of individual packet switch nodes interconnected by 56 kbps leased lines (Class A) [Com88].

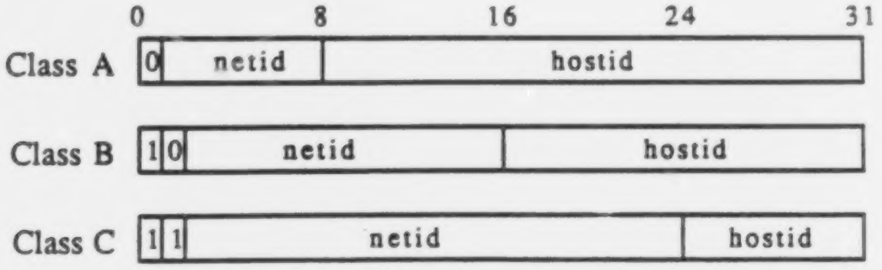
BBN: (Bolt, Beranek, and Newman, Inc.) A Cambridge, MA, company responsible for development, operation, and monitoring of the ARPAnet and Internet core gateway systems, CSnet Coordination and Information Center (CIC), and NSFnet Network Service Center (NNSC) [Com88].

BITnet: (Because It's Time Network) A low cost, low speed network connecting over 200 universities nationally and internationally. It consists mostly of IBM mainframe computers connected by 9600 bps leased lines. It provides mainly electronic mail services [Com88].

bridge: A node connected to two or more administratively indistinguishable but physically distinct subnets, that automatically forwards datagrams when necessary, but whose existence is not known to other hosts.

CICnet: The Committee on Institutional Cooperation (CIC) received a grant from the NSF to establish a regional network connecting the Big-Ten universities in the mid-west plus the University of Chicago (Class B).

Classes of Internet Networks: Each Internet Address consists of a network identifier and host identifier component. There are three classes of networks with their address formats shown below.



Thus, the main difference between the network classes is that Class A networks are few in number but can support a large number of hosts (between 2^{16} and 2^{24} hosts), and Class B networks are more in number and support an intermediate number of hosts (between 2^8 and 2^{16} hosts), while Class C networks are many in number but can support a small number of hosts (up to 2^8 hosts) [Com88].

CSnet: (Computer Science Network) A logical network of institutions involved in computer science research offering mail delivery service using three physical networks: PhoneNet, X25Net and ARPAnet [Com88].

CSO: Computing Services Office of the University of Illinois at Urbana-Champaign.

domain: Part of the Internet naming hierarchy. A domain name consists of a sequence of labels separated by dots. Domain name servers are software servers (often executed on a dedicated processor) that supplies the name-to-address translation by mapping the domain name to Internet address. For example, the domain name *uxc.cso.uiuc.edu* refers to the host named *uxc* which is directly administered by the sub-domain *cso*, the Computer Sevices Office (CSO). This is in turn under the domain *uiuc*, the University of Illinois at Urbana Champaign, and this university is assigned to the top-level domain *edu*, which represents all the educational institutions.

dotted devimal notation: A representation of a 32-bit integer that consists of four 8-bit numbers written in base 10, with periods (dots) separating them [Com88]. IP addresses are commonly written in this form.

EGP: (Exterior Gateway Protocol) Protocol used by a gateway in one autonomous system to announce the Internet address of networks in that autonomous system to a gateway in another autonomous system [Com88].

FTP: (File Transfer Protocol) The Internet standard protocol for transferring files from one machine to another. It is an Application Layer program that uses the Telnet and TCP protocols [Com88].

gated: (Gateway Daemon) A program that runs under Unix 4.3 BSD that allows a gateway to collect information from within one autonomous system using RIP, HELLO or some other IGP, and announce these routes to another autonomous system using EGP.

hardware address: The low level address used by physical networks. Each type of network hardware has its own addressing scheme [Com88]. For example, ethernet uses a vendor-assigned 48-bit hardware address, while the ProNET-10/80 token rings uses smaller user-assigned hardware addresses.

Hello: The protocol used by a group of cooperative, trusting switches to allow them to discover minimal delay routes [Com88].

ICMP: (Internet Control Message Protocol) A part of the IP used by gateways and hosts to send reports of problems about datagrams back to the source of the datagram [Com88].

IGP: (Interior Gateway Protocol) Generic term applied to any protocol used to propagate network reachability and routing information within an autonomous system. Although there is no Internet standard, RIP is the most popular [Com88].

Internet: The collection of networks and gateways, including the ARPAnet, MILnet and NSFnet, that use the TCP/IP protocol suite and function as a single, cooperative virtual network [Com88]. The Internet is composed of over 900 subnets provides universal connectivity with services like remote login, file transfer and electronic mail. It offers three levels of network services:

i. unreliable, connectionless packet delivery
ii. reliable, full duplex connection-oriented delivery
iii. Application Layer services like electronic mail using (i) and (ii)

Internet Address: The Internet address is a unique 32-bit address assigned to each host on the Internet and is used in all communication with that host. It is normally written dotted decimal notation. For example, the Internet address 128.174.5.50 refers to the host number 50 on subnet 5 of the Class B network 128.174 [Com88].

IP: (Internet Protocol) The Internet standard protocol that defines the Internet datagram as a unit of information passed across the Internet and provides the basis for the Internet connectionless, best-effort packet delivery service [Com88].

mail bridge/gateway: A machine that connects two or more electronic mail (often dissimilar) systems and transfers mail messages among them. Mail gateways usually capture capture an entire mail message, reformats it according to the rules of the destination mail system, and then forwards the message [Com88].

MILnet: (Military Network) Originally part of ARPAnet, MILnet was partitioned in 1984 to make it possible for military installations to have a reliable network service while ARPAnet continued to be used for research [Com88].

MRnet: (Minnesota Research network) This is a regional network connecting the Minnesota Supercomputing Center and the northern mid-west universities (Class B).

Nameserver: This protocol provides a machine-based routine for translating a host name to an Internet address.

NSFnet: (National Science Foundation network) Consists of the cross country backbone, mid-level networks and supercomputer consortia networks that have all been started with NSF funds [Com88].

power cycled: This involves switching the power off and then on again after a while to reset a device. This ensures that all state information is cleared and that the device is restored to its initial state. Software resets do not always completely restore a device to its initial state.

proNET-10/proNET-80: Commercially available local area network product of Proteon Inc. that operates at 10 Mbps (proNET-10) or 80 Mbps (proNET-80) using token passing ring technology [Com88].

regional network: Term applied to mid-level networks which are funded by the National Science Foundation. They operate autonomously but connect to the NSFnet backbone [Com88]. These connect groups of universities in the same geographical area.

RFC: (Request For Comments) The name of a series of notes that contain surveys, measurements, ideas, techniques, and observations, as well as proposed and accepted Internet protocols and standards. These are available across the Internet [Com88].

RIP: (Routing Information Protocol) The protocol used by 4.3BSD Unix systems to exchange routing information among a (usually small) set of computers. Typically, the participating machines all attach to a single local area network [Com88].

rlogin: (remote login) The service offered by 4.3BSD Unix systems that allows users of one machine to connect to other Unix systems across the Internet as if their terminals connected to the remote machine directly. Rlogin offers the same service as Telnet except that it also passes information about the user's environment (eg. terminal type) to the remote machine [Com88].

routed: A program that runs under 4.3BSD Unix to propagate routes among machines on a local area network. It uses the RIP protocol [Com88].

router: A special-purpose, dedicated computer between two or more administratively distinct networks or subnets, to which hosts send datagrams to be forwarded. The router forwards the packets from one network or subnet to the other.

SMTP: (Simple Mail Transfer Protocol) The Internet standard protocol for transferring electronic mail messages from one machine to another [Com88].

SNMP: (Simple Network Management Protocol) This is a protocol by which management information for a network element may be inspected or altered.

subnet: One or more physical networks forming a subset of an Internet network. A subnet is explicitly identified in the Internet address by means of its subnet number [MoP85]. Thus, an Internet address of a subnetted network takes the form:

<network-number> <subnet-number> <host-number>

T1: A serial line with an overall data rate of 1.544 Mbps. It multiplexes 24 data streams into slots of 8-bits each with an extra bit per frame for synchronization [BeG87]

TCP: (Transmission Control Protocol) The Internet standard transport level protocol that provides reliable, full duplex, connection-oriented service. TCP allows a process on one machine to send a stream of data to a process on another. TCP uses IP to transmit information across the Internet [Com88].

Telnet: The Internet standard protocol for remote terminal connection service. Telnet allows a user at one site to interact with a remote timesharing system at another site as if the user's terminal connected directly to the remote machine [Com88].

TFTP: (Trivial File Transfer Protocol) The Internet standard protocol for file transfer with minimal capability and overhead. TFTP depends on the unreliable, connectionless datagram delivery service, UDP, and is used by machines like diskless workstations to bootstrap themselves [Com88].

UDP: (User Datagram Protocol) The Internet standard protocol that allows an application program on one machine to send a datagram to an application program on another machine. UDP uses IP to deliver datagrams [Com88].

UIRnet: University of Illinois Regional network (Class B).

UIUCnet: University of Illinois at Urbana-Champaign campus network (Class B).

USA-CERL: US Army Corps of Engineers, Construction Engineering Research Laboratory network (Class B).

UUCP: (Unix to Unix Copy Program) Allows one Unix system to copy files to or from another Unix system over a single (usually dialup) link. It is often used loosely to refer to Unix mail transfer [Com88].

115

REFERENCES

[AKW88] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *The AWK Programming Language*, Addison-Wesley Publishing Company, Reading, MA, 1988.

[All89] Bob Allan, "What Managers Really Need To Know About LANs", *IEEE Network Magazine*, Vol 3 No 6, July 1989.

[BeG87] Dimitri P. Bertsekas, and Robert G. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[BeY89] Lawrence Bernstein, and Christine M. Yuhas, "How Technology Shapes Network Management", *IEEE Network Magazine*, Vol 3 No 4, July 1989.

[BrP87] Robert T. Braden, and Jonathan B. Postel, "Requirements for Internet Gateways, RFC-1009", USC Information Sciences Institute, November 1988.

[CaA88] L. N. Cassel, and P. D. Amer, "Management of Distributed Measurement over Interconnected Networks", *IEEE Network Magazine*, Vol 2 No 2, March 1988.

[Cal90] Ross W. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments, Draft RFC", Digital Equipment Corp, January 1990.

[CaQ87] Smoot Carl-Mitchell, and John S. Quarterman, "Using ARP to Implement Transparent Subnet Gateways, RFC-1027", Network Information Center (NIC), SRI International, Menlo Park, CA, October 1987.

[CCB88] Robert N. Cronk, Paul H. Callahan, and Lawrence Bernstein, "Rule-based Expert Systems for Network Management and Operations: An Introduction", *IEEE Network Magazine*, Vol 2 No 5, September 1988.

[Cer88] Vinton G. Cerf, "Internet Activities Board (IAB) Recommendation for the Development of Internet Network Management Standards, RFC-1052", Network Information Center, SRI International, Menlo Park, CA, April 1988.

[CFS89] Jeffrey D. Case, Mark S. Fedor, Martin L. Schoffstall, and James R. Davin, "Simple Network Management Protocol (SNMP), RFC-1098", Network Information Center, SRI International, Menlo Park, CA, April 1989.

[Com88] Douglas E. Comer, *Internetworking with TCP/IP: Principles, Protocols and Architecture*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[CrG85] B. Croft and J. Gilmore, "BOOTP - UDP Bootstrap Protocol, RFC-951", Network Information Center (NIC), SRI International, Menlo Park, CA, August 1985.

[CST88] Jitti Chiarawongse, Mandyam M. Srinivasan, and Toby J. Teorey, "Performance Analysis of a Large Interconnected Network by Decomposition Techniques", *IEEE Network Magazine*, Vol 2 No 4, July 1988.

[DCF87] James R. Davin, Jeffrey D. Case, Mark S. Fedor, and Martin L. Schoffstall, "A Simple Gateway Monitoring Protocol, RFC-1028", Network Information Center (NIC), SRI International, Menlo Park, CA, November 1987.

[DCF88] James R. Davin, Jeffrey D. Case, Mark S. Fedor, and Martin L. Schoffstall, "Introduction to the Simple Gateway Monitoring Protocol", *IEEE Network Magazine*, March 1988.

[FLN88] Metin Feridun, Melisse Leib, Marian H. Nodine, and James C. Ong, "ANM: Automated Network Management System", *IEEE Network Magazine*, Vol 2 No 2, March 1988.

[FMM84] Ross Finlayson, Timothy Mann, Jeffrey Mogul, and Marvin Theimer, "A Reverse Address Resolution Protocol, RFC-903", Network Information Center (NIC), SRI International, Menlo Park, CA, June 1984.

[GAD85] GADS (Gateway Algorithms and Data Structures Task Force), "Toward an Internet Standard Scheme for Subnetting, RFC-940", Network Information Center (NIC), SRI International, Menlo Park, CA, April 1985.

[Ged89] Carolyn A. Gedney, "Campus Connectivity: The Long and Winding Road", *UIUCnet: The Campus Network*, Vol 2 No 3, University of Illinois, August 1989.

[Hor84] C. Hornig, "A Standard for the Transmission of IP Datagrams over Ethernet Networks, RFC-894", Symbolics, April 1984.

[ISO87a] Information Processing Systems - Open Systems Interconnection, "Protocol for Providing the Connectionless-mode Network Service, DIS-8473, RFC-994", International Organization for Standardization, January 1987.

[ISO87b] Information Processing Systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1), IS-8824", International Organization for Standardization, December 1987.

[ISO87c] Information Processing Systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), IS-8825", International Organization for Standardization, December 1987.

[Kro89] Edward M. Krol, "The Hitchhikers Guide to the Internet, RFC-1118", University of Illinois at Urbana-Champaign, September 1989.

[Mar88] Todd E. Marques, "A Symptom Driven Expert System for Isolating and Correlating Network Faults", *IEEE Communications Magazine*, Vol 26 No 3, March 1988.

[McR88] Keith McCloghrie, and Marshall T. Rose, "Management Information Base for Network Management of TCP/IP-based internets, RFC-1066", Network Information Center, SRI International, Menlo Park, CA, August 1988.

[MeZ90] Douglas Meyer, and George Zobrist, "TCP/IP vs OSI: The Battle of the Network Standards", *IEEE Potentials*, February 1990.

[Mog84a] Jeffrey Mogul, "Internet Subnets, RFC-917", Network Information Center (NIC), SRI International, Menlo Park, CA, October 1984.

[Mog84b] Jeffrey Mogul, "Broadcasting Internet Datagrams, RFC-919", Network Information Center (NIC), SRI International, Menlo Park, CA, October 1984.

[Mol88] J. F. Mollenauer, "Standard for Metropolitan Area Networks", *IEEE Communications Magazine*, Vol 26 No 4, April 1988.

[MoP85] Jeffrey Mogul, and Jonathan B Postel, "Internet Standard Subnetting Procedure, RFC-950", Network Information Center (NIC), SRI International, Menlo Park, CA, August 1985.

[Pau86] L. Pau, "Survey of Expert Systems for Fault Detection, Test Generation, and Maintenance", *Expert Systems*, Vol 3 No 2, April 1986.

[Plu82] David C. Plummer, "Ethernet Address Resolution Protocol, RFC-826", Network Information Center (NIC), SRI International, Menlo Park, CA, November 1982.

[Pos84] Jonathan B. Postel, "Multi-LAN Address Resolution, RFC-925", USC Information Sciences Institute, October 1984.

[Pos88] Jonathan B. Postel, "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, RFC-1042", USC Information Sciences Institute, February 1988.

[RBS88] Sameh Rabie, Timothy Bult, Vicky Snarr-Carter, and Brian Cavan, "DRESS: Diagnostic Repair Expert System Shell", *IEEE International Conference on Communication (ICC) 1988*, Vol 3.

[Red89] Shyamala Reddy, "NSFnet Today: A New Implementation of a Vast Research Network", *LAN Magazine*, June 1989.

[RiS87] Donna Ritter, and Marilyn Seale, "A Multi-Purpose, Distributed LAN Traffic Monitoring Tool", *IEEE Network Magazine*, Vol 1 No 3, July 1987.

[RoM88] Marshall T. Rose, and Keith McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets, RFC-1065", Network Information Center, SRI International, Menlo Park, CA, August 1988.

[Ros89] Marshall T. Rose, "Management Information Base for Network Management of TCP/IP-based internets, Draft RFC (Obsoletes RFC-1066)", Network Information Center (NIC), SRI International, Menlo Park, CA, December 1989.

[RRS88] Sameh Rabie, Andrew Rau-Chaplin, and Taro Shibahara, "DAD: A Real-Time Expert System for Monitoring of Data Packet Networks", *IEEE Network Magazine*, Vol 2 No 5, September 1988.

[SDF89] Martin L. Schoffstall, James R. Davin, Mark S. Fedor, and Jeffrey D. Case, "SNMP over Ethernet, RFC-1089", Network Information Center, SRI International, Menlo Park, CA, February 1989.

[Sng89] Dennis C. H. Sng, "External Connections of the University of Illinois at Urbana-Champaign Campus Network", UIUC Department of Computer Science Technical Report No. UIUCDCS-R-89-1540, August 1989.

[Soh87] Michael Soha, "A Distributed Approach to LAN Monitoring Using Intelligent High Performance Monitors", *IEEE Network Magazine*, Vol 1 No 3, July 1987.

[Sta87] William Stallings, *Handbook of Computer Communications Standard, Vol 2: Local Network Standards*, Macmillan Publishing Company, New York, 1987.

[Ste89] Lou Steinberg, "Draft Memo for Managing Asynchronously Generated Alerts", IETF Alert Management Working Group, December 1989.

[SuZ88] Mark T. Sutter, and Paul E. Zeldin, "Designing Expert Systems for Real-Time Diagnosis of Self-Correcting Networks", *IEEE Network Magazine*, Vol 2 No 5, September 1988.

[Tan88] Andrew S. Tanenbaum, *Computer Networks*, 2nd Ed, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[WaB89] Unnikrishnan S. Warrier, and Lawrence Besaw, "Common Management Information Services and Protocol over TCP/IP (CMOT), RFC-1095", Network Information Center, SRI International, Menlo Park, CA, April 1989.

[ZTD88] Wei-Dong Zhan, S Thanwastien, and L M L Delcambre, "SIMNETMAN: An Expert Workstation for Designing Rule-Based Network Managament Systems", *IEEE Network Magazine*, Vol 2 No 5, September 1988.

BIBLIOGRAPHIC DATA SHEET	**1. Report No.** UIUCDCS R-90-1595	**2.**	**3. Recipient's Accession No.**
4. Title and Subtitle NETWORK MONITORING AND FAULT DETECTION ON THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN CAMPUS COMPUTER NETWORK			**5. Report Date** April 1990 **6.**
7. Author(s) Dennis Cheng-Hong Sng			**8. Performing Organization Rept. No.** UIUCDCS -R-1595
9. Performing Organization Name and Address University of Illinois Department of Computer Science 1304 W. Springfield Avenue Urbana, Illinois 61801			**10. Project/Task/Work Unit No.** **11. Contract/Grant No.**
12. Sponsoring Organization Name and Address			**13. Type of Report & Period Covered** technical **14.**
15. Supplementary Notes			

16. Abstracts

The University of Illinois at Urbana-Champaign (UIUC) has a large campus computer network serving a community of about 20,000 users. With such a large network, it is inevitable that there is a wide variety of technologies co-existing in a multi-vendor environment. Effective network monitoring tools can help in monitoring traffic and link usage, as well as for detecting faults and correcting them before serious degradation or loss of service to the user community occurs.

This paper describes a network monitoring and fault detection tool that we have developed and which is currently in operation at the UIUC. This tool makes use of the Simple Network Management Protocol (SNMP) to collect data from the various network elements monitored and processes the data into reports for the network administrator. We will also give a few examples of faults that have been detected with this tool.

17. Key Words and Document Analysis. 17a. Descriptors

Computer Networks, Network Fault Detection, Network Fault Diagnosis, Network Fault Isolation, Network Monitoring, Polling, CIC net and UIUC net.

18. Availability Statement	**19. Security Class (This Report)** UNCLASSIFIED	**21. No. of Pages** 119
	20. Security Class (This Page) UNCLASSIFIED	**22. Price**

FORM NTIS-35 (10-70) USCOMM-DC 40329-P71